Quality Control in Crowdsourcing based on Fine-Grained Behavioral Features

WEIPING PEI, Colorado School of Mines, USA
ZHIJU YANG, Colorado School of Mines, USA
MONCHU CHEN, Appen, USA
CHUAN YUE, Colorado School of Mines, USA

Crowdsourcing is popular for large-scale data collection and labeling, but a major challenge is on detecting low-quality submissions. Recent studies have demonstrated that behavioral features of workers are highly correlated with data quality and can be useful in quality control. However, these studies primarily leveraged coarsely extracted behavioral features, and did not further explore quality control at the fine-grained level, i.e., the annotation unit level. In this paper, we investigate the feasibility and benefits of using fine-grained behavioral features, which are the behavioral features finely extracted from a worker's individual interactions with each single unit in a subtask, for quality control in crowdsourcing. We design and implement a framework named Fine-grained Behavior-based Quality Control (FBQC) that specifically extracts fine-grained behavioral features to provide three quality control mechanisms: (1) quality prediction for objective tasks, (2) suspicious behavior detection for subjective tasks, and (3) unsupervised worker categorization. Using the FBQC framework, we conduct two real-world crowdsourcing experiments and demonstrate that using fine-grained behavioral features is feasible and beneficial in all three quality control mechanisms. Our work provides clues and implications for helping job requesters or crowdsourcing platforms to further achieve better quality control.

CCS Concepts: • **Human-centered computing → Collaborative and social computing design and evaluation methods**; **Empirical studies in collaborative and social computing**.

Additional Key Words and Phrases: crowdsourcing; quality control; behavior analysis

ACM Reference Format:
Weiping Pei, Zhiju Yang, Monchu Chen, and Chuan Yue. 2021. Quality Control in Crowdsourcing based on Fine-Grained Behavioral Features. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW2, Article 442 (October 2021), 28 pages. https://doi.org/10.1145/3479586

1 INTRODUCTION

With the increasing popularity and use of crowdsourcing platforms (such as Mechanical Turk of Amazon [28] and Figure Eight of Appen [1]) in the AI era [2, 36], quality control has become more critical and challenging than before due to the heterogeneous nature of workers [12, 22] and the existence of malicious workers or attackers [3, 6, 7, 20]. Much research effort has been made in these years to develop quality control approaches in crowdsourcing. One popular approach is to compare a worker's submissions against a set of labeled high-quality data (i.e., the Gold Standard) [39]. Another widely employed approach is to introduce redundancy, which involves assigning the same subtask to a number of workers and then inferring the consensus label by using aggregation

Authors' addresses: Weiping Pei, Colorado School of Mines, Golden, USA, weipingpei@mines.edu; Zhiju Yang, Colorado School of Mines, Golden, USA, zhijuyang@mines.edu; Monchu Chen, Appen, Sunnyvale, USA, mochen@appen.com; Chuan Yue, Colorado School of Mines, Golden, USA, chuanyue@mines.edu.

2573-0142/2021/10-ART442 $15.00
https://doi.org/10.1145/3479586

methods, such as Majority Voting [21] and the Dawid-Skene model [5]. However, constructing the gold standard requires a considerable amount of human effort, and massive redundancy makes a data annotation task costly. Moreover, these two approaches are often not applicable to tasks such as opinion polls and surveys that are subjective in nature.

Recently, there is an increasing interest in estimating data quality by analyzing worker behaviors [6, 18, 34]. This approach tracks worker activities (e.g., mouse clicks and keypresses) during the task execution, and then estimates the quality of submissions by analyzing workers' behavioral data. Compared to the gold standard and the redundancy analysis approaches, this approach can help reduce costs (in terms of time and money) and be feasible for scaling up.

1.1 Coarse vs. Fine-Grained Behavioral Features

On a crowdsourcing platform, a requester would publish a task that could be broken down into multiple subtasks. Each individual subtask can be further split into multiple units, each of which is usually a simple atomic annotation or response from a worker. Therefore, the annotations collected from workers could be divided into different levels of granularity: ***unit level***, ***subtask level***, and ***task level***. Specifically, we define ***unit data*** (i.e., unit level data) as a single annotation or response provided by a worker in a subtask; we define ***subtask data*** (i.e., subtask level data) as all annotations provided by a worker in a subtask; we define ***task data*** (i.e., task level data) as all annotations provided by a worker in an entire task. It is worth to highlight that having multiple units in a subtask has become very common in many types of crowdsourcing tasks such as image segmentation, emotion recognition, text annotation, relevance judgment, and survey with the corresponding units being segmented objects in an image, utterances in a dialogue, annotated words in a paragraph, query-document pairs, and survey questions, respectively; meanwhile, different subtasks of a task may consist of different numbers of units.

Correspondingly, the quality of the crowdsourced data can be estimated or controlled at different granularities. In this paper, we use the term ***fine-grained*** to represent the unit level, and use the term ***coarse-grained*** to represent subtask and task levels. The quality of the unit data is specific to each individual annotation or response unit (e.g., in terms of correctness), while the quality of the subtask data or task data is ratio of high-quality to all annotations or responses in a subtask or the entire task, respectively. We define ***coarse-grained behavioral features*** as the behavioral features coarsely extracted from a worker's overall interactions with each entire subtask, such as the total completion time and the number of mouse clicks in an entire subtask. We define ***fine-grained behavioral features*** as the behavioral features finely extracted from a worker's individual interactions with each single unit in a subtask, such as the time spent and the number of mouse clicks on each single unit. These two types of behavioral features can both be extracted when a job requester or a crowdsourcing platform embeds JavaScript code to the task webpages. However, they differ in the granularities at which they are extracted, and consequentially at which they inform us about a worker's behavior and data quality.

1.2 Fine-grained Behavior-based Quality Control

Existing behavior-based quality control studies such as [6, 18, 34] mainly focus on estimating the overall quality of a subtask by extracting and analyzing coarse-grained behavioral features. Unfortunately, ***coarse-grained behavioral analysis can lead to the inclusion of low-quality data, exclusion of high-quality data, and/or manipulation by malicious workers*** because it ignores a worker's time-varying behaviors among annotation units in the same subtask. For example, a worker may carefully perform the first half annotation units in a subtask, but become sloppy and try to complete the rest annotations quickly due to decreasing interest. This results in

high-quality for the first half annotations in a subtask but low-quality for the rest. And a coarse-grained behavioral analysis would label the entire subtask either as high-quality thus including a sizable proportion of low-quality annotations into the dataset, or as low-quality thus wasting many high-quality and valuable annotations. Meanwhile, a coarse-grained behavioral analysis would be evaded by malicious workers because they can easily emulate a legitimate worker's overall behavior in a subtask. For example, they would spend enough time on some annotations in a subtask, but use tools to automatically complete the rest annotations.

Towards addressing these limitations in coarse-grained behavioral analysis, we ***investigate the feasibility and benefits of using fine-grained behavioral features for quality control in crowdsourcing***. Especially, we consider two major uses of behavioral features for quality control in crowdsourcing. One is ***quality estimation***, which includes mechanisms such as ***quality prediction*** that often builds models to directly predict the quality of a certain piece of data, and ***suspicious behavior detection*** that often uses heuristics or rules to detect suspicious (or abnormal) behaviors of workers. These two mechanisms can help a job requester or a crowdsourcing platform estimate the quality of the submitted data, exclude low-quality data, and even prompt workers in real-time for more careful data submission. The other use (with the corresponding mechanism) is ***worker categorization***, which is performed either manually or automatically. The categorization results can help a job requester or a crowdsourcing platform identify different types of workers for proactive worker pre-selection, better worker-task matching, and malicious worker exclusion. We hypothesize that in both uses or the three mechanisms, leveraging fine-grained behavioral features can contribute to better quality control.

In terms of quality estimation, we emphasize that it should be performed differently for objective tasks vs. subjective tasks because ground-truths typically exist for the former while not for the latter. Prior studies mainly focused on utilizing coarse-grained behavioral features in objective tasks (Section 2). Instead, we consider leveraging fine-grained behavioral features to perform quality prediction for objective tasks and perform suspicious behavior detection for subjective tasks. In terms of worker categorization, we emphasize that unsupervised categorization is desirable because crowd workers are so diverse in terms of their backgrounds, intentions, and behaviors. A few prior studies manually categorized workers based on different criteria (Section 2). Researchers in a recent study [6] extracted coarse-grained behavioral features and built supervised machine learning models to predict worker types at the subtask level. However, it can be expensive to collect sufficient labeled data for different types of workers; meanwhile, the authors did not analyze the time-varying behaviors of a worker across the subtasks within a task. Instead, we explore to use fine-grained behavioral features to categorize workers at the whole task level in an unsupervised manner, which can be beneficial for the overall quality control such as task assignment based on worker types and identification of malicious workers.

In particular, we design and implement a framework named Fine-grained Behavior-based Quality Control (***FBQC***). This framework aims to assist job requesters and crowdsourcing platforms for better quality control at different granularities by specifically extracting and analyzing fine-grained behaviors of workers. It consists of three components. The first component, *fine-grained behavior monitoring*, is designed to collect workers' fine-grained behavioral traces by specifically logging their interactions with the individual units of each subtask; it also collects workers' coarse-grained behavioral traces at the subtask level. The second component, *feature extraction at multiple granularities*, is designed to extract workers' fine-grained behavioral features (referred to as ***UB*** features, i.e., Unit Behavioral features) and coarse-grained behavioral features (referred to as ***TB*** features, i.e., subTask Behavioral features) by using the traces logged in the first component; it also extracts the task attributes of individual units and subtasks as ***UA*** (Unit Attribute) features and ***TA*** (subTask Attribute) features, considering their potential effects on data quality. The third component, *multiway*

quality control, is designed to analyze the extracted features and perform (1) quality prediction for objective tasks at both the fine-grained and coarse-grained levels, (2) suspicious behavior detection for subjective tasks at both the fine-grained and coarse-grained levels, and (3) unsupervised worker clustering at the task level.

To evaluate the effectiveness of FBQC, we conducted two real-world crowdsourcing experiments on the popular Figure Eight crowdsourcing platform of Appen [1]. First, we designed an image task of *visual object detection*, which is an objective task, to evaluate FBQC's quality prediction performance. The experimental results based on 258 workers demonstrate the feasibility and benefits of using fine-grained features to predict data quality at the fine-grained level. Specifically, our model performs the best by leveraging both UB and UA features; it predicts the quality of unit data with a 28.7% reduction in Mean Square Error (MSE) compared to a baseline model, and identifies low vs. high quality unit data with a 70.0% accuracy, which is 9.2% higher than the baseline model. Meanwhile, the experimental results also validate the usefulness of leveraging fine-grained behavioral features for predicting data quality at higher levels. For example, leveraging TB, UB, and TA features together, our model performs the best and achieves 9.4% to 18% improvement in identifying low vs. high quality subtask data compared to prior works [8, 34]. These results also imply that not only the worker behavior but also the task characteristics should be considered in data quality prediction. Second, we designed a text task of *textual emotion recognition*, which is a subjective task, to evaluate FBQC's suspicious behavior detection performance on 427 workers. Manual examination results show that our method achieves a 90.0% accuracy in detecting suspicious behavior at the fine-grained level. Meanwhile, we find that majority of the suspicious behaviors are related to tool usages. Finally, we build unsupervised clustering models to categorize workers at the task level. The evaluation results demonstrate that fine-grained behavioral features perform better than coarse-grained behavioral features in distinguishing different types of workers. To facilitate the adoption of our framework, we implemented a JavaScript library[1] that can be easily used by job requesters to obtain fine-grained behavioral features from different types of task webpages.

To summarize, we make the following major contributions in this paper:

(1) We propose to explore quality control in crowdsourcing by specifically extracting and analyzing fine-grained behavioral features.
(2) We design and implement the FBQC framework that a) performs quality prediction for objective tasks and suspicious behavior detection for subjective tasks at different granularities, and b) clusters different types of workers at the whole task level in an unsupervised manner.
(3) We design and conduct two crowdsourcing experiments to validate the feasibility and benefits of using fine-grained behavioral features in quality control, and demonstrate the effectiveness of our FBQC framework.

The rest of this paper is organized as follows. Section 2 reviews the related work and further justifies the uniqueness of our work. Section 3 describes the design of the FBQC framework and experiments. Section 4 evaluates the effectiveness of our framework in quality prediction and suspicious behavior detection. Section 5 explores and analyzes task-level worker categorization. Section 6 further discusses our study from multiple perspectives including (1) workers' tool usages in our tasks, (2) generalizability, deployability, and scalability of the FBQC framework, and (3) limitations and future work. Section 7 concludes this paper.

2 RELATED WORK

Quality Control in Crowdsourcing. There are three major approaches to quality control in crowdsourcing: gold standard [39], redundancy analysis [5, 21], and worker behavior analysis [8,

[1]https://github.com/weipingpei/Crowdsourcing-Quality-Control-FBQC

13, 34]. Our work resides in the third approach, which is a non-intrusive approach that can help reduce costs (in terms of time and money) and be feasible for scaling up. We focus on reviewing this category of work.

Rzeszotarski et al. [34] first proposed to utilize workers' behavioral traces, referred to as task fingerprints, to predict the quality of subtask data. They conducted experiments on word classification, image tagging, and reading comprehension tasks, and showed that behavioral features could be used to estimate the quality of subtask data and detect potential cheaters. Since then, worker behavior analysis has been widely explored in different types of tasks such as relevance judgment [8], video quality-of-experience assessment [27], and object segmentation [11]. Moreover, Han et al. [9] demonstrated the better performance of worker behavior analysis in quality prediction compared to the conventional method that is based on workers' historical performance signals. To further improve the performance of worker behavior analysis, Kazai et al. [18] proposed to train a classifier based on gold judges' behavioral signals which are generated by trained professional judges. Their experimental results demonstrated that gold behaviors help improve the accuracy of worker quality prediction. While in these studies behavioral features have been demonstrated to be highly correlated with data quality, behavior-based quality control may fail in tasks that are too simple to capture sufficient behavioral features. Suzuki et al. [37] tackled this problem by requesting a worker to perform extra operations in a task.

These prior studies primarily focus on estimating the overall quality of coarse-grained subtask data and task data by extracting and analyzing coarse-grained behavioral features. In contrast, we specifically extract and analyze fine-grained behavioral features; meanwhile, our study involves the quality estimation of not only coarse-grained data but more importantly fine-grained data (i.e., unit data). Although studies such as [8, 11] seem to be close to the fine-grained quality prediction, their proposed features are still extracted at the subtask level. In addition, prior studies did not consider the situation that the subtasks of many types of tasks often contain varying numbers of units. We consider this situation and conduct experiments on two crowdsourcing tasks that naturally have varying numbers of units in each subtask.

Crowd Worker Categorization. Considering the heterogeneous nature of workers in crowdsourcing, categorizing workers based on their behaviors or characteristics can be helpful in identifying the desired workers to improve quality control. Worker categorization can be performed either manually or automatically.

Prior studies proposed different definitions of worker types based on different criteria. Kazai et al. [17] defined five worker types based on the task completion time and historical quality of annotations: spammer (malicious workers delivering unacceptable responses), sloppy (workers completing a task as fast as possible), incompetent (workers lacking of skills or competencies), competent (efficient and effective workers), and diligent (workers completing a task carefully). Vuurens et al. [38] introduced four worker types simply based on the quality of annotations: diligent workers, sloppy workers, random spammers, and uniform spammers. To gain further insights into malicious activities, Gadiraju et al. [7] manually categorized untrustworthy workers into five types based on their responses to a survey: ineligible workers, fast deceivers, rule breakers, smart deceivers, and gold standard preys.

Recently, workers' behavioral traces have been used for worker categorization. Gadiraju et al. [6] proposed to combine behavior, motivation, and performance data to typecast workers. They built supervised machine learning models to predict worker types based on the behavioral features extracted at the subtask level. In contrast, we leverage fine-grained behavioral features to cluster crowd workers by using unsupervised machine learning techniques. Meanwhile, they did not analyze the time-varying behaviors of a worker across the subtasks within a task. Instead, we categorize workers at the whole task level to capture such behaviors, which can be beneficial for

the overall quality control such as task assignment based on worker types and identification of malicious workers.

Table 1 summarizes the comparison with the aforementioned related studies on behavior analysis in crowdsourcing from the feature usage, quality control mechanism, and task design perspectives. In addition to behavioral features, this table also considers task attribute features which are taken into account in our framework and will be detailed in Section 3.2. While related studies primarily leverage features extracted at the subtask level, our FBQC framework specifically takes behavioral features at the unit level into account. Meanwhile, our FBQC framework is applicable for quality estimation in subjective tasks, and our study considers the situation that subtasks often contain multiple and varying numbers of units.

Table 1. Comparison with Related Studies on Behavior Analysis in Crowdsourcing.

	Behavioral Features			Task Attribute Features		Quality Control Mechanism			Task Design
	Unit Level	Subtask Level	Task Level	Unit Level	Subtask Level	Quality Estimation		Worker Categorization	# Units Per Subtask
						Objective Task	Subjective Task		
[34]		✓				✓			multiple
[18]		✓	✓			✓			multiple
[27]		✓				✓			multiple
[8]		✓				✓			one
[11]		✓			✓	✓			one
[6]		✓						✓	multiple
FBQC	✓	✓	✓	✓	✓	✓	✓	✓	multiple & varying

3 DESIGN OF THE FRAMEWORK AND EXPERIMENTS

3.1 Framework Design and Its Rationale

To explore whether finer details provided by fine-grained behavioral features can facilitate the quality control in crowdsourcing, we propose a framework named FBQC, representing Fine-grained Behavior-based Quality Control. The goal of this framework is to help its users including job requesters and crowdsourcing platforms effectively control the quality of crowdsourced data at different granularities by specifically extracting and analyzing fine-grained behaviors of workers.

As shown in Figure 1, our FBQC framework consists of three components. The first component, *fine-grained behavior monitoring*, is designed for collecting workers' fine-grained behavioral traces by specifically logging their interactions with the individual units of each subtask; it also collects workers' coarse-grained behavioral traces at the subtask level. With the logged traces from the first component, the *feature extraction at multiple granularities* component extracts workers' fine-grained behavioral (i.e., ***UB*** or Unit Behavioral) features and coarse-grained behavioral (i.e., ***TB*** or subTask Behavioral) features. Meanwhile, considering the potential effects of task attributes on data quality, this component also extracts ***UA*** (i.e., Unit Attribute) features and ***TA*** (i.e., subTask Attribute) features by analyzing task attributes at the unit level and the subtask level, respectively. We will soon introduce how the extracted UB and TB features can be aggregated to derive statistical behavioral features at the subtask and task levels.

Extracted features will be used in the last component, *multiway quality control*, which consists of two sub-components and provides three mechanisms for quality control. The *quality prediction & suspicious behavior detection* sub-component provides the quality prediction mechanism for objective tasks and the suspicious behavior detection mechanism for subjective tasks at both the fine-grained and coarse-grained levels. The *task-level worker clustering* sub-component provides the worker categorization mechanism in an unsupervised manner at the task level. Based on the output or feedback from the first sub-component, a job requester or a crowdsourcing platform can estimate the quality of the submitted data, exclude low-quality data, and even prompt workers

in real-time for more careful data submission. Based on the output or feedback from the second sub-component, a job requester or a crowdsourcing platform can identify different types of workers for proactive worker pre-selection, better worker-task matching, and malicious worker exclusion.

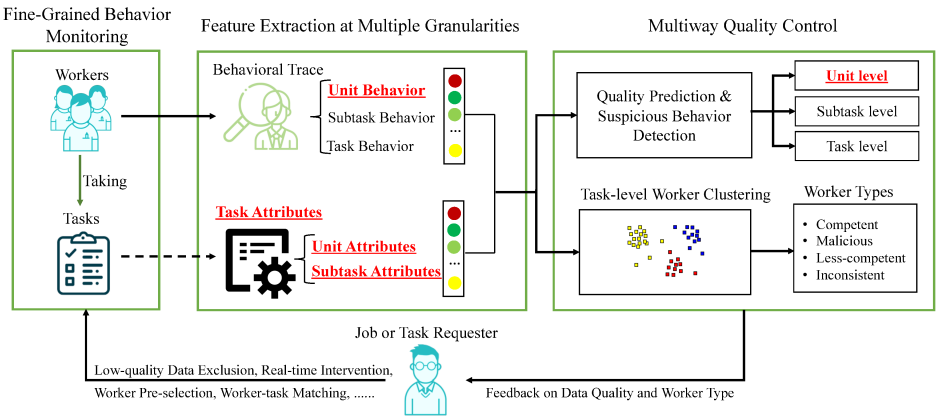


Fig. 1. The Architecture of Our Proposed Fine-grained Behavior-based Quality Control (FBQC) Framework.

3.1.1 Fine-Grained Behavior Monitoring. This component is designed to collect fine-grained behavioral traces from workers during task execution. Data collection occurs on a worker's web browser, e.g., by using JavaScript code embedded to the task webpages. The collected behavioral data can be simply sent to a server that belongs to either the crowdsourcing platform or the job requester. We implement this component using JavaScript and the JQuery library [16] to monitor workers' activities and log their interactions with the task including mouse movements, mouse clicks, keypresses, and scrolls, etc. In general, annotations provided by workers are submitted to the crowdsourcing server through HTML *<input>* elements. The attributes of each *<input>* element are typically changed by a worker's interactions. For example, checking a radio button via a mouse would trigger mouse click events and change the *checked* and *focused* attributes of the element. Thus, by analyzing the buttons that have been checked, the positions of the mouse cursor, and so on, we can programmatically identify the units that a worker has spent some time and effort on. It is worth highlighting that this fine-grained behavioral data collection capability can be easily added in an automatic (e.g., adding event handlers for all *<input>* and other important elements on a webpage) or semi-automatic (e.g., adding event handlers to web elements via a wizard) manner by either a job requester or a crowdsourcing platform vendor. To facilitate the adoption of our framework, we share our implemented JavaScript library that can be easily used by job requesters to obtain fine-grained behavioral features from their task webpages.

3.1.2 Feature Extraction at Multiple Granularities. This component extracts behavioral features at different granularities. For the UB features, we split all the logged events in a subtask into different groups corresponding to the different units in the subtask; we then extract UB features based on the grouped events per unit, such as the total number of mouse clicks in a unit. For the TB features, we extract them based on the events per subtask, such as the total number of mouse clicks in a subtask. UA and TA features are extracted based on the task attributes (such as difficulty and complexity pertaining to a specific task type) that are measured at the unit level and the subtask level, respectively. In the next subsection, we will describe the detailed UB, UA, TB, and TA features extracted in our real-world experiments.

UB and UA features can be directly used to perform the fine-grained level (i.e., unit level) quality control. Correspondingly, TB and TA features can be directly used to perform the subtask level quality control. However, since different subtasks may contain different numbers of diverse units, it would be beneficial to capture such information and further incorporate UB features into the subtask level quality control. This requires us to aggregate UB features of all units in a subtask to derive statistical behavioral features, for which we used the mean, median, standard deviation, minimum, and maximum statistical calculations in our implementation. We use $\boldsymbol{UB}^*$ to denote these statistical UB features that are aggregated at the subtask level. We will use UB^* features together with TB and TA features for the subtask level quality control.

Similarly, different workers may complete different numbers of diverse subtasks. To capture the time-varying behaviors of a worker across the subtasks within a task, we can aggregate UB features of all units completed by the worker in the task to derive statistical UB features in a similar way (i.e., statistical calculations); we can also aggregate TB features of all subtasks completed by the worker in the task to derive statistical TB features in a similar way. We use $\boldsymbol{UB}^\#$ to denote these statistical UB features that are aggregated at the task level, and use $\boldsymbol{TB}^\#$ to denote these statistical TB features that are aggregated at the task level. We consider $\text{UB}^\#$ and $\text{TB}^\#$ features as task behavioral features of a worker, and will use them together for the task level quality control.

3.1.3 Multiway Quality Control. With features extracted at different granularities, this last component provides three mechanisms to help job requesters or crowdsourcing platforms control data quality at different levels. Its *quality prediction & suspicious behavior detection* sub-component aims to estimate the quality of data at different granularities. However, we emphasize that quality estimation should be performed differently for objective tasks vs. subjective tasks. Objective tasks are supposed to have the ground-truths, and annotations that are close to the ground-truths can be considered and measured as high-quality. Whereas, subjective tasks permit many reasonable annotations and there is no or no solid ground-truth; therefore, reliable or non-suspicious (i.e., less likely coming from malicious or irresponsible workers) data can be considered as high-quality. To enable quality estimation and corresponding control for different types of tasks, our framework therefore performs quality prediction for objective tasks and performs suspicious behavior detection for subjective tasks. The *task-level worker clustering* sub-component constructs unsupervised models for task-level worker categorization. It identifies different clusters of similar workers based on their task behavioral features. We expect that our task-level worker categorization could help capture the overall behavior of a worker for a given type of task, thus facilitating the effective assignment of different tasks to different workers and the identification of malicious workers.

3.2 Experimental Design and Setup

To evaluate the utility and effectiveness of our FBQC framework, we conducted two real-world experiments on the popular Figure Eight crowdsourcing platform of Appen [1].

3.2.1 Task Design. Text and image annotations are still the main types of tasks performed on crowdsourcing platforms [15]. Besides, our framework is designed to be applicable to both objective tasks and subjective tasks. Therefore, we design an image task of *visual object detection* as the objective task, and a text task of *textual emotion recognition* as the subjective task in our experiments.

Visual Object Detection. This task is designed based on the public Open Image dataset [19], and we focus on object detection, which is one of the fundamental problems in visual recognition. Specifically, in our task, each subtask presents an image and workers are asked to draw bounding boxes to locate all people in the image as shown in Figure 2a. In our task description, we provided a detailed instruction on using the bounding box tool, and exemplified some high-quality annotations. In this case, each created bounding box corresponds to a piece of unit data, while all created

bounding boxes in an image correspond to the subtask data. In this task, we used 200 sampled images from the Open Image dataset. Those images contain varying numbers of people ranging from 3 to 10, and we gathered annotations from 10 distinct workers for each image. To estimate the required time to complete a subtask, we conducted a pilot study and found that the median time for an individual to annotate an image was less than 45 seconds. Thus, we set the payment rate at 10 USD cents for per image annotation to compensate workers. The projected hourly wage is (0.10 USD/image×3600 sec/hr) / (45 sec/image) = 8.0 USD/hr, which was above the required minimum wage rate of 7.25 USD/hr [10] in the United States.

Textual Emotion Recognition. This task is designed based on the public dataset MELD [4], which is popularly used in natural language processing for sentiment analysis. In our task, each subtask presents a dialogue with several utterances as shown in Figure 2b, and workers are asked to check one of the seven options that best describes the emotion of each utterance. Here an utterance corresponds to a unit. In this task, we used 420 sampled dialogues from the MELD dataset with varying numbers of utterances ranging from 4 to 24, and gathered annotations from 10 distinct workers for each dialogue. The median time for an individual to complete a dialogue in our conducted pilot study was less than 60 seconds. Thus, we set the payment rate at 15 USD cents for per dialogue completion to compensate workers. The projected hourly wage is (0.15 USD/dialogue × 3600 sec/hr) / (60 sec/dialogue) = 9.0 USD/hr, which was also above the required minimum wage rate in the United States.

(a) The partial user interface of a subtask in the image task.

(b) The partial user interface of a subtask in the text task.

Fig. 2. Two subtask examples in the image task and the text task, respectively.

Collected Data. For both tasks, no test questions are used to screen workers and all subtasks are open to all workers in the USA. To have more workers involved, we only allow each worker to complete at most 50 subtasks, and do not allow a worker to complete the same subtask twice. Besides, the minimum time per subtask is set to 10 seconds by the platform, so workers who submit a subtask within 10 seconds would be rejected automatically and we do not have their data. We capture workers' behavioral data only on our task webpages and with the IRB approval. Table 2 lists the number of completed units, completed subtasks, and workers in the two tasks, while Figure 3 shows the distribution of workers by the number of their completed subtasks. We can see that many workers only completed one subtask in a task. Specifically, 64 workers (23.7%) only completed one image subtask, and 101 workers (24.5%) only completed one text subtask. Meanwhile, we find that 10 workers completed the maximum allowed 50 dialogues in the text task and 47 workers made submissions in both tasks.

Table 2. Summary of the Collected Data including the Number of Completed Units and Subtasks.

Task Type	# images or dialogues	# units	# subtasks	# workers
Image	200	10,984	1,948	258
Text	460	49,395	4,451	427

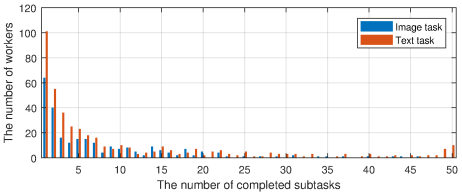


Fig. 3. Distribution of workers by the number of their completed subtasks.

3.2.2 Extracted Features. From the logged events, our framework extracts both fine-grained and coarse-grained features.

Fine-Grained Features. They consist of workers' behavioral features extracted on each unit and the attributes of each unit in a subtask as shown in Table 3. For the UB (Unit Behavioral) features, we not only extract time-related features such as *time_on_unit* and mouse events related features such as *total_[X]_events*, but also extract behavioral features regarding how a worker creates an annotation, such as the number of times that a worker changes an option (*num_change_annotation*) in the text task for potential annotation revision. For the UA (Unit Attribute) features, we may have different types of features for different types of tasks. For example, in our text task, we leverage basic textual features such as the number of words in an utterance, considering that a longer utterance may need more time for a worker to read and annotate. As reviewed in Section 2, prior studies did not specifically extract and analyze fine-grained behavioral (i.e., UB) features. Correspondingly, they did not extract and analyze UA features.

Table 3. Fine-Grained Features Extracted for Each Unit

Feature Type	Feature Name	Description
Unit Behavioral (UB) Features	time_on_unit	Time spent on a unit task, i.e., a bounding box in the image task or an utterance in the text task.
	total_[X]_events	The number of logged events of type X for a bounding box or an utterance where X could be one in {create, remove} in the image task, or {clicks, keypresses, checks} in the text task.
	num_change_annotation	The number of times that a worker deletes a bounding box or changes an option.
	events_around_annotation	The number of logged events immediately around the annotation action for a unit, including clicks, keypresses, movements, etc.
	movement_speed_unit	The mean, median, and standard deviation of mouse movement speed within the created bounding box or the utterance.
	speed_around_annotation	The mean, median, and standard deviation of mouse movement speed before/after creating a bounding box or selecting an option for an utterance.
Unit Attribute (UA) Features	unit_attributes	The attributes of a unit, i.e., the size and entropy of a bounding box in the image task, or the number of words and prepositions of an utterance in the text task.

Coarse-Grained Features. Table 4 lists coarse-grained features that we extracted at the subtask level. We extract TB (subTask Behavioral) features as in prior studies (Section 2), such as the time spent on each subtask (*time_on_subtask*). We also consider TA (subTask Attribute) features such as the total number of units in a subtask. As described in the subsection above, for the coarse-grained quality control, the extracted UB and TB features can be aggregated to derive statistical behavioral features at the subtask and task levels. That is, UB^* features will be derived and used together with TB and TA features for the subtask level quality control; $UB^\#$ and $TB^\#$ features will be derived and used together for the task level quality control. It is important to note that although prior studies (Section 2) used TB features, they did not extract unit-related TA features and did not derive UB^* and $UB^\#$ features simply because they did not explore the unit level concept to specifically extract fine-grained features.

4 QUALITY PREDICTION AND SUSPICIOUS BEHAVIOR DETECTION

Quality estimation for the purpose of quality control can be achieved by our framework in different ways for subjective tasks and objective tasks. Since objective tasks are considered as having the

Table 4. Coarse-Grained Features Extracted for Each Subtask

Feature Type	Feature Name	Description
subTask Behavioral (TB) Features	time_on_subtask	Total time spent on a subtask, i.e., an image or a dialogue.
	total_[X]_events	The number of logged events of type X for a subtask where X could be one in {create, remove} in the image task, or {clicks, keypresses, checks} in the text task.
	time_on_instruction	Time spent by a worker on reading the task instruction before starting the first unit.
	tBeforeInput	Time taken by a worker before creating the first bounding box or choosing the first option in a subtask.
subTask Attribute (TA) Features	subtask_attributes	The attributes of a subtask, i.e., the size and entropy of an image in the image task, or the number of utterances of a dialogue in the text task.

ground-truths that are less influenced by personal feelings or interpretations, quantitatively measuring the quality of workers' submissions is both feasible and highly desirable. Correspondingly, we propose to construct machine learning models to provide quality prediction for objective tasks at both the fine-grained and coarse-grained levels. Different from objective tasks, there is no or no solid ground-truth for subjective tasks, and obtaining reliable (i.e., less likely coming from malicious or irresponsible workers) submissions is more feasible and meaningful. Thus, we propose to detect suspicious behaviors of workers for subjective tasks. This detection will provide a useful indication of the submission quality and reliability for potentially removing suspicious submissions. In this section, we evaluate the effectiveness of our FBQC framework on quality prediction and suspicious behavior detection for the aforementioned image task and text task, respectively.

4.1 Quality Prediction for the Image Task

The image task we designed and described in Section 3 is largely an objective task. Therefore, we build machine learning models to predict the data quality for this image task, and evaluate their performance at the fine-grained and coarse-grained levels.

4.1.1 Fine-Grained Level Prediction. The goal of the fine-grained level quality prediction is to predict the quality of a unit, which is an annotated bounding box in the image task. We leverage fine-grained features (i.e., UB and UA features listed in Table 3) to build supervised machine learning models, and evaluate their performance for the fine-grained level quality prediction. In the image task, we use the Intersection over Union (IoU) [33] to measure the quality, which is defined as the size of the intersection between the submitted bounding box and the ground-truth bounding box divided by the size of their union. The quality can be predicted in two ways: using regression models that estimate the continuous IoU values, and using classification models that identify low vs. high quality units. Specifically, regression models output the continuous IoU values for units in which a higher predicted IoU value indicates a higher quality, while classification models directly output whether the unit data is high-quality or low-quality based on some predefined criterion such as a qualification threshold. In our experiments, we define low-quality units as those with IoU values falling below the average IoU value across all bounding boxes, and use this definition for labeling the units. A similar definition of low-quality can be found in [18].

Metrics and Models. Regarding the evaluation metrics, we report the *mean-squared error (MSE)* for the regression models and *accuracy* for the classification models. For regression, we train and test a support vector (machine) regression (SVR) model and a random forest regression (RFR) model. For classification, we train a support vector (machine) classifier (SVC) and a random forest classifier (RFC). Note that our framework is not limited to these models and other models could also be used. We use a grid search for hyperparameters in the candidate sets as follows: (1) the maximum number of features is searched from *(2, 3, log2)* and the number of estimators is searched from *(2, 4, 6, 8, 10, 20)* for the random forest related models (RFR and RFC); (2) the regularization parameter is searched from *(1, 5, 10)* for the support vector machine related models (SVR and SVC).

Five-fold cross validation is applied and we partition the data via images such that all unit data from the same image would appear only in the training data or in the testing data. Specifically, in each iteration of the five-fold cross validation, 80% (i.e., 160 out of the total 200) images are used for training and 20% images are used for testing. Here for the unit level prediction, we train models using all collected unit data (with their UB and UA features) of the training images, and evaluate models on all collected unit data (with their UB and UA features) of the testing images.

Since we are the first to investigate quality prediction at the unit level, there are no literatures studying this problem for comparison. Thus, we compare our models with two baseline models for regression and classification, respectively. The baseline model for regression always outputs a single value, which is the mean of IoU values of all units (i.e., bounding boxes) in the training images. The baseline model for classification always outputs the label of the majority class of all units in the training images. That is, if the majority of units in the training images are high-quality, this model outputs "high-quality"; if the majority of units in the training images are low-quality, this model outputs "low-quality". While appearing to be straightforward, these two baseline models are still strong because they leverage the overall knowledge of the training images. Considering that training data and testing data should come from the same distribution in the supervised learning (as we do here), this overall knowledge of the training images enable the two baseline models to easily outperform a random guessing model. For example, in our experiments, the baseline model for classification always outputs "high-quality" for predicting the quality of each piece of unit data in the testing images, and can achieve a 60.8% testing accuracy as shown below in Table 5.

Overall Results and Analysis. We present the fine-grained (i.e., unit level) quality prediction results using different types of features as listed in Table 5. We can see that by combining UB (Unit Behavioral) features with UA (Unit Attribute) features, which corresponds to UB&UA in the table, the RFR model achieves the best result (0.0184 MSE), which is 28.7% lower than the baseline model (0.0258 MSE) in predicting IoU values. Moreover, the RFC model with UB&UA performs the best for identifying low vs. high quality units, which achieves a 70.0% accuracy. These results confirm that it is both feasible and beneficial to leverage fine-grained behavioral features to predict data quality at the fine-grained level. They also imply that not only the worker behavior on the units but also the characteristics of the units should be considered in the fine-grained level quality prediction. Meanwhile, using unit behavioral features exclusively performs better than using unit attribute features exclusively, which indicates that the former is more relevant than the latter in helping predict data quality for the image task. While we report results on those traditional learning models, one might wonder whether a modern neural network approach could achieve better performance. To investigate this, we experimented with the LSTM model [14], but did not observe a better performance (e.g., at most a 67.3% accuracy in classification) likely due to the limited training data size.

Table 5. Unit Level Quality Prediction (UB - Unit Behavioral Features, UA - Unit Attribute Features)

Model Type	Baseline	SVR/SVC			RFR/RFC		
Features	-	UB	UA	UB&UA	UB	UA	UB&UA
Regression (MSE×100)	2.58	2.29	2.86	2.23	1.94	2.15	**1.84**
Classification (Accuracy)	60.8%	69.6%	63.6%	69.9%	69.5%	61.2%	**70.0%**

Predictive Features. To gain further insights into the effects of the selected features on building a model to predict quality, we analyzed the correlation between each individual feature and the predicted quality. Table 6 shows the features with positive and negative correlation coefficients with the annotation quality predicted by the trained RFR model[2]. We can see that *time_on_unit*

[2]This model achieves the best performance.

is positively correlated with the quality, which suggests that the more time a worker spends on creating a bounding box, the higher the annotation quality would be. We can also observe that unit attribute features *bbox_size* and *bbox_entropy* (i.e., the size and entropy of a created bounding box) also have positive correlation with the quality, indicating that it is easier to get a higher IoU value for a larger target in an image and for a target with richer details. We found that most of the mouse speed related features are negatively correlated with quality. This is also reasonable because lots of adjustments before a worker starts or finishes drawing a bounding box for ensuring high quality would slow down the mouse speed. Overall, these results suggest that the quality of a unit is well reflected by a worker's annotation behavior for the unit and by the attributes of the unit.

Table 6. Features with Positive and Negative Correlation Coefficients

Features with Positive Coef.		Features with Negative Coef.	
Feature	Coef.	Feature	Coef.
time_on_unit	0.460	median_speed_before_end_point	-0.481
bbox_size	0.252	median_speed_before_start_point	-0.473
events_around_end_point	0.224	mean_speed_before_start_point	-0.453
bbox_entropy	0.223	mean_speed_before_end_point	-0.351
events_around_start_point	0.218	std_speed_before_start_point	-0.316
		median_speed_bbox	-0.187
		canny	-0.175
		mean_speed_bbox	-0.169
		std_speed_before_end_point	-0.155
		std_speed_bbox	-0.155

4.1.2 Coarse-Grained Level Prediction. The coarse-grained level quality prediction includes the prediction of the subtask data quality and the task data quality. Here, subtask data correspond to all bounding boxes provided by a worker for a given image, while task data correspond to all bounding boxes for all images provided by a worker in the entire image task. The quality of subtask data or task data is measured as the average IoU value of all involved bounding boxes, respectively. Similar to that in Section 4.1.1, we train regression models to estimate the continuous average IoU values, and train classification models to identify low-quality vs. high-quality data, but now the models are working at the subtask and task levels.

Metrics and Models. We use the same metrics, train similar models, apply five-fold cross validation, and create similar baseline models as in the fine-grained level prediction in Section 4.1.1. However, here we partition the data by unique images for the subtask level quality prediction, but by unique workers for the task level quality prediction. The baseline model for regression at the subtask level (or task level) always outputs a single value, which is the mean of the average IoU values of all subtask data (or task data) in the training images. The baseline model for classification at the subtask level (or task level) always outputs the label of the majority class of all subtask data (or task data) in the training images. Similarly, these baseline models are straightforward yet still strong for the same reason that they leverage the overall knowledge of the training images. For example, in our experiments, the baseline model for classification at the subtask level (or task level) also always outputs "high-quality" for predicting the quality of an entire subtask data (or an entire task data of a worker) in the testing images, and can achieve a 66.3% (or 72.0%) testing accuracy as shown below in Tables 7 and 8. Moreover, at the subtask level, we further compare our models with those in [8, 34], which are two representative behavior-based prior studies for predicting the quality of the subtask data. Specifically, we implement their proposed models: Decision Tree with Aggregate Features (DT-AF), Random Forest with Aggregate Features (RF-AF), and Random Forest with Sequence Features (RF-SF).

At the task level, we further compare our models with a widely used quality control approach, *Gold Standard*, which estimates the quality of unlabeled data by comparing collected data against a set of labeled data (i.e., gold questions with the ground-truths). We analyze that if the first subtask completed by a worker were used as a gold question[3] to predict the quality of the rest subtasks completed by the worker, what would be prediction performance. Specifically, the Gold Standard model for regression always uses the average IoU value of all bounding boxes completed by a worker for the first image (i.e., the gold question or gold image) as the predicted IoU value for all the rest images completed by the same worker; similarly, the Gold Standard model for classification always uses the quality (either high-quality or low-quality) of a worker's annotation of the first image as the predicted quality for all the rest images completed by the same worker. Note that although these two models are based on a hypothetical scenario, their prediction performance indeed represents their capability if they were in use in practice. Also note that it is typically inappropriate to directly use one worker's response to a gold question to predict the data quality of another worker. Therefore, our comparison with the Gold Standard approach is only at the task level for each individual worker.

Overall Results and Analysis. Table 7 presents the subtask level quality prediction results using different types of features and different models. Due to their comparable performance to RFR and RFC, we do not present the results of SVR and SVC in the table. We can see that the baseline model for regression (0.0416 MSE) performs worse than all other behavior-based regression models. Moreover, we can observe that by combining fine-grained behavioral features (here more specifically the UB* statistical features) and coarse-grained features (TB and TA features), the RFR model achieves the best performance with 0.0076 MSE for predicting average IoU values. This result is 49.3% lower than that of RF-SF, which achieves the best performance among models in the prior studies. With the same combination of features, the RFC model obtains the highest prediction accuracy 83.4% with 9.4% higher than that of RF-AF, which achieves the best performance among models in the prior studies for identifying low vs. high quality subtask data. These results validate the usefulness of fine-grained behavioral features and the benefits of including subtask characteristics in the subtask level quality prediction. Moreover, using fine-grained behavioral features exclusively performs better than using coarse-grained behavioral features exclusively in all our models, which further confirms the strong relevance between the fine-grained behavioral features and the data quality.

Table 7. Subtask Level Quality Prediction (TB - subTask Behavioral Features, UB - Unit Behavioral Features, TA - subTask Attribute Features. Here UB* features are statistical features derived from UB features of all units in a subtask as described in Section 3.1.2.)

Model Type	Baseline	DT-AF [8, 34]	RF-AF [8]	RF-SF [8]	RFR/RFC				
Features	-	-	-	-	TB	UB*	TB&UB*	TA	TB&UB*&TA
Regression (MSE×100)	4.16	3.1	1.8	1.5	1.57	0.89	0.90	1.07	**0.76**
Classification (Accuracy)	66.3%	65.4%	74.0%	73.9%	67.8%	83.1%	82.7%	75.8%	**83.4%**

Table 8 presents the task level quality prediction results using different types of features. We can see that baseline and Gold Standard models perform worse than all our models. Meanwhile, by using both unit behavioral features (here more specifically the UB# statistical features) and subtask behavioral features (here more specifically the TB# statistical features), our RFR model achieves the lowest MSE (0.009 MSE) in regression and our RFC model achieves the highest accuracy

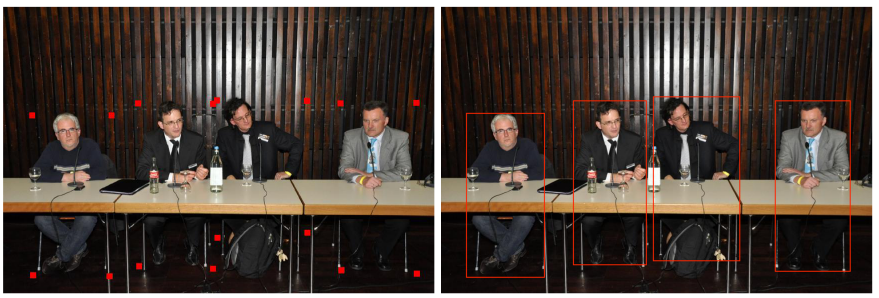
[3] Appen [1] usually places a test subtask (i.e., a gold question) as the first subtask to a worker.

(84.7%) in classification. These results demonstrate that fine-grained behavioral features (here more specifically the $UB^{\#}$ statistical features) are also helpful in the task level quality prediction.

Table 8. Task Level Quality Prediction (TB - subTask Behavioral Features, UB - Unit Behavioral Features. Here $TB^{\#}$ features are statistical features derived from TB features of all subtasks in a task, and $UB^{\#}$ features are statistical features derived from UB features of all units in a task as described in Section 3.1.2.)

Model Type	Baseline	Gold Standard	SVR/SVC			RFR/RFC		
Features	-	-	$TB^{\#}$	$UB^{\#}$	$TB^{\#}$&$UB^{\#}$	$TB^{\#}$	$UB^{\#}$	$TB^{\#}$&$UB^{\#}$
Regression (MSE×100)	3.44	1.60	1.58	1.41	1.53	1.26	0.89	**0.90**
Classification (Accuracy)	72.0%	74.6%	78.5%	83.1%	82.4%	81.2%	82.0%	**84.7%**

Case Study. Figure 4 shows the annotations of an image provided by a worker and the corresponding ground-truth annotations. Instead of using a bounding box to annotate each person in the image, this worker used four points[4] to locate each person. It is obvious that this worker did not follow the instruction and got a low IoU value (almost 0) for each person, resulting in low-quality annotations for this image. However, points in this image were not drawn randomly and in fact they are very close to the vertexes of the ground-truth bounding boxes. We believe that this worker did try to locate each person precisely but simply misunderstood the task instruction. Although these are low-quality annotations, creating those 16 points actually results in similar coarse-grained behavioral features as in high-quality annotations. Thus, annotations of this image are misclassified as high-quality by the RFC model that only uses $TB^{\#}$ features. In contrast, the RFC models that use $UB^{\#}$ features are able to correctly identify the annotations of this image as low-quality. Therefore, by extracting fine-grained behavioral features, we are able to have finer details regarding a worker's behavior on each unit for better quality control.



(a) Annotations Provided by a Worker. (b) Ground-truth Annotations.

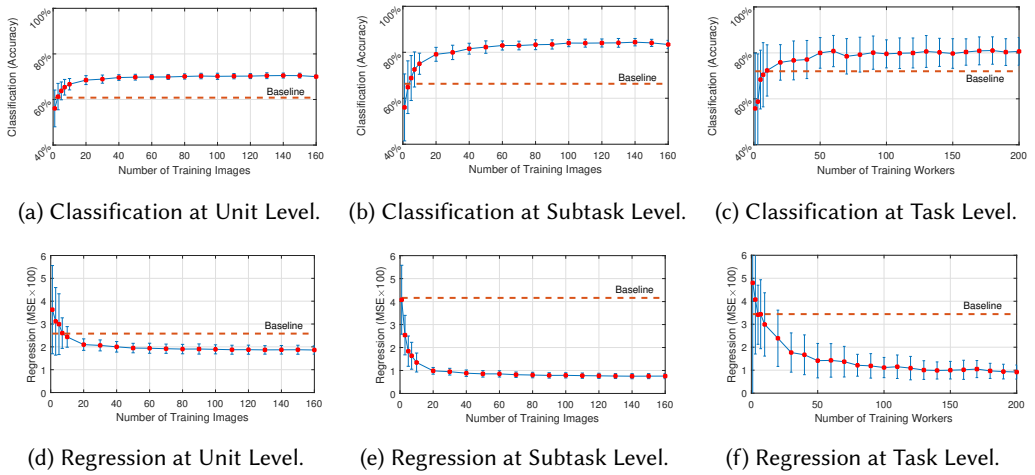
Fig. 4. Annotations Provided by a Worker and the Corresponding Ground-truth Annotations.

4.1.3 Guideline on Using FBQC for Quality Prediction. For job requesters who have visual object detection or other objective tasks such as text annotation and receipt transcription, they simply need to follow the procedure that we used in our image task experiments to perform either fine-grained or coarse-grained level quality prediction. In short, they need to: (1) enable the collection of fine-grained behavioral features (e.g., by including our JavaScript library into their task webpages), (2) launch a task with a small portion of subtasks to create a training dataset with the collected behavioral features and the data quality ground-truths (e.g., IoU values in our image task), (3) train regression or classification models in a supervised manner, (4) use the trained models to predict

[4]The worker only provided a point with 1 pixel × 1 pixel. We have enlarged each point to make it clear.

the quality of the large portion of subtasks and their units, and finally (5) remove low-quality submissions (e.g., at either the unit level or the subtask level) to improve the overall quality of the collected annotations or responses.

We acknowledge that a training dataset with the ground-truth labels must be constructed for our quality prediction models because they use supervised machine learning methods. We demonstrated in the experiments above that our models can achieve a good performance with a relatively small training dataset, and the same dataset can be used to train several models at different levels. We also highlighted that once a model is trained, a job requester can use it to predict the quality of much more subtasks and their units. To further analyze the impact of the training dataset size on the prediction performance, we trained our models using smaller datasets of different sizes. Figure 5 presents the classification accuracies of the RFC models and the regression MSEs of the RFR models with varying numbers of training images or workers.



(a) Classification at Unit Level. (b) Classification at Subtask Level. (c) Classification at Task Level.

(d) Regression at Unit Level. (e) Regression at Subtask Level. (f) Regression at Task Level.

Fig. 5. Classification accuracy and regression MSE with varying numbers of training images or workers at three different levels. Red dots and blue sticks on those curves indicate the mean and standard deviation values, respectively.

Recall that in our five-fold cross validation experiments at the unit level (Section 4.1.1) and the subtask level (Section 4.1.2), we used 80% (i.e., 160 out of the total 200) images and their annotations for training. Specifically, the models at the subtask level are trained with 160 images × 10 subtasks/image = 1600 subtasks, the models at the unit level are trained with around 9,000 units (images have varying numbers of units). Meanwhile, the models at the task level (Section 4.1.2) are trained with all images annotated by 206 workers (80% of the 258 workers). Now from the unit level results shown in Figures 5a and 5d, we can observe that with only 40 training images, the classification model still achieves a good accuracy (69.6%) and the regression model obtains a low MSE (0.02). Similarly at the subtask level as shown in Figures 5b and 5e, our models still achieve more than 80% accuracy for classification and obtain less than 0.01 MSE for regression by only using 40 training images. At the task level as shown in Figures 5c and 5f, our models still achieve a high accuracy (80%) for classification and a low MSE (0.012) for regression by only using the data of 100 workers (a very small portion of workers on a popular crowdsourcing platform) for training. These results imply that in practice a job requester may just need to build a small labeled dataset in order to train a model of good performance, and one reason is that we used traditional learning instead of neural network models as analyzed in Section 4.1.1. This investment on a small dataset

will be well worth it especially if the requester's actual task contains a much larger number of subtasks and units to be labeled.

4.2 Suspicious Behavior Detection for the Text Task

The text task we designed and described in Section 3 is largely a subjective task that is difficult to build a model for predicting the data quality. Therefore, we consider detecting suspicious behaviors in a heuristic manner. Specifically, we design rules for detecting suspicious behaviors of workers at the fine-grained level, with the detection results being usable for excluding suspicious annotations at different levels.

4.2.1 Fine-Grained Level Detection. By obtaining behavioral features on a per unit basis, we are able to identify suspicious (or abnormal) behaviors because some normal behaviors of a worker on a unit are expectable in advance. For example, we expect that a mouse click event or a keypress event will be triggered and the interaction duration should be sufficient if a worker manually checks or unchecks a radio button in the text task. For the text task, our framework automatically detects suspicious behaviors using the following rules: (1) the time spent on a unit (*time_on_unit*) is less than a threshold t_r; (2) there is no mouse click or keypress observed in a unit; (3) none of radio buttons in a unit has been put on focus during the subtask execution. If all conditions in these three rules are fulfilled, our framework detects a worker's behavior on this unit as suspicious. It is worth to note that the conditions in the rules may vary depending on the task, for example, other types of elements instead of radio buttons may appear in a unit.

Overall Results and Analysis. With different time threshold t_r values, this rule-based method detected different percentage of suspicious behaviors over the total 49,395 completed units as shown in Table 9. A job requester has the choices to apply a stricter standard or a looser one, depending on which direction would be more appropriate for the corresponding task.

Table 9. Suspicious Behavior Detection Results over the Total 49,395 Completed Units based on Different Time Threshold t_r Values

t_r = 0.1 second		t_r = 0.2 second		t_r = 0.3 second		t_r = 0.4 second		t_r = 0.5 second	
Suspi.	Non-Suspi.	Suspi.	Non-Suspi.	Suspi.	Non-Suspi.	Suspi.	Non-Suspi.	Suspi.	Non-Suspi.
47.8%	52.2%	52.1%	48.8%	52.9%	47.1%	54.1%	45.9%	54.9%	45.1%

We can observe that a great percentage (over 50% on average) of the completed units are detected as suspicious in the text task. This result is surprising but understandable because such a simple and subjective text task provides many opportunities for workers to make irresponsible submissions by using tools. To verify the effectiveness of our rule-based method, we randomly sampled and carefully inspected 200 suspicious units and 200 non-suspicious units identified with $t_r = 0.5$. Especially, we inspected a worker's behavioral traces in a whole subtask, the utterance content, and the created annotations to identify behaviors that are obviously suspicious. Table 10 shows the overall performance and the confusion matrix of our method for detecting suspicious behaviors on the 400 sampled units. We can see that our method achieves a 90.0% accuracy, which confirms the feasibility and usefulness of detecting suspicious behaviors at the fine-grained level.

Case Study. During manual inspection, we found that majority of the detected suspicious annotation units are impossible to be made by humans assuming an utterance must be read before being annotated. Take Figure 6 as an example, all 14 radio buttons for 14 utterances are checked with "neutral" within 0.1 second after a "right_click" mouse event is triggered, resulting in a series of consecutive "check_option" events which are suspicious. No relevant events such as mouse movements on any individual unit are observed because in this example those radio buttons were

Table 10. Performance of Fine-Grained Level Suspicious Behavior Detection

(a) Confusion Matrix.

		Manual Inspection	
		Suspi.	Non-Suspi.
Automated Detection	Suspi.	186	14
	Non-Suspi.	26	174

(b) Overall Performance.

Accuracy	Precision	Recall	F1 score
90.0%	93.0%	87.7%	90.3%

[event_id]	[event type]	[event subtype]	[mouse position]	[timestamp]
[88]	mouse_click	right_click	(168, 597)	30.299
[89]	mouse_enter	utterance3_margin	(168, 597)	32.411
[90]	mouse_move	utterance3_margin	(1023, 636)	32.414
[91]	check_option	utterance0_option_neutral	(1023, 636)	32.462
[92]	check_option	utterance1_option_neutral	(1023, 636)	32.471
[93]	check_option	utterance2_option_neutral	(1023, 636)	32.485
.....				
[103]	check_option	utterance12_option_neutral	(1023, 636)	32.551
[104]	check_option	utterance13_option_neutral	(1023, 636)	32.555
[105]	mouse_move	utterance3_margin	(993, 661)	32.565

Fig. 6. Logged Events of Suspicious Behaviors. A series of consecutive "check_option" events are suspicious and are highlighted in red.

not manually checked by a human; this assures us that some tools must be in use for this low-quality or irresponsible submission. In another example which also has 14 utterances, we found that all utterances are checked with "neutral" when the page is loaded without capturing a series of consecutive "check_option" events as shown in Figure 6. The worker spent sufficient time to carefully revise the first three utterances and then directly scrolled down the webpage to submit the annotations. This example illustrates the fact that even in the same subtask, it is possible that partial units are annotated suspiciously while the rest units are annotated normally. By specifically capturing workers' behavioral features at the unit level, our framework is able to identify suspicious units in a subtask and then excludes them for quality control.

As shown in Table 10a, there are 14 false positives (i.e., detected as suspicious but verified as non-suspicious) and 26 false negatives (i.e., detected as non-suspicious but verified as suspicious). By analyzing those false positives, we found that our method was confused by some efficiency-driven tool-using cases as suspicious. It is worth to note that we should not always consider tool-using cases as suspicious. For example, we observed that some efficient workers used tools to select a default option and then made changes. In terms of false negatives, some units manually completed by a worker without utilizing tools are detected as non-suspicious but verified as suspicious because the corresponding responses are largely invalid. For example, a worker manually annotated the units in some obvious patterns, such as checking "anger" for all the utterances in one subtask while checking "joy" for all the utterances in another subtask.

4.2.2 Coarse-Grained Level Detection. As we have seen above, our framework is able to detect suspicious behaviors at the fine-grained level with high accuracy. This is important and beneficial because suspicious annotations on individual units can then be confidently removed to improve the overall data quality. We should note that it can be straightforward to extend our fine-grained suspicious behavior detection to coarse-grained subtask and task levels by aggregating fine-grained detection results. For example, a job requester could set a suspicious rate threshold to filter out the data at the entire subtask or even task level. The suspicious rate is defined as the ratio of units performed with suspicious behaviors to all units in either a subtask or a task. In other words, if a job

requester considers the suspicious rate in a subtask performed by a worker as too high, this entire subtask submission of the worker can be discarded; if the suspicious rates in multiple subtasks performed by a worker are too high, the entire task submission of the worker can be discarded. These flexible decisions are up to job requesters or crowdsourcing platforms (if they would remove suspicious submissions on behalf of job requesters), so we do not further discuss them in this paper.

A Comparison with Using the Attention Check Mechanism. Attention check questions [29] have been widely used in quality control for subjective tasks such as surveys [26, 30]. However, the attention check mechanism has a limited capability and may not often be an appropriate choice for at least two reasons. First and foremost, we should realize that each attention check question is an interaction unit by itself, and its use is at the subtask level. That is, based on a worker's answer to an attention check question, the entire subtask data would typically be accepted for the correct answer or discarded for an incorrect answer. In other words, the attention check mechanism is not able to achieve what our fine-grained level detection can achieve (as just presented in Section 4.2.1). Second, we should note that even at the subtask level, it is less applicable to use attention check questions in some types of tasks such as the textual emotion recognition task [4, 32]; this is because inserting an attention check question between utterances would break the continuity of the entire dialogue and induce an interruption that may affect workers' judgments. On crowdsourcing platforms, it is indeed unusual to identify the use of attention check questions in such types of data labeling tasks. Moreover, this mechanism could be vulnerable because attention check questions can be automatically answered by machine learning models [31].

To further explain the incapability of the attention check mechanism on detecting the quality of the unit data, we analyzed the consistency of the unit data quality in each of the 4,451 completed subtasks in our text task. We found that 35.03% of these completed subtasks contain mixed suspicious and non-suspicious unit data, 35.16% of them only contain non-suspicious unit data, and 29.81% of them only contain suspicious unit data. This result indicates that workers' efforts and behaviors indeed vary a lot within a subtask. Let us consider a hypothetical scenario by assuming that the attention check mechanism was in use in our task. When an attention check question in a subtask is incorrectly answered, all the high-quality unit data in the subtask would be discarded thus wasted; this undesirable situation would occur to all those 35.03% mixed subtasks and even to those 35.16% subtasks that only contain non-suspicious unit data. On the other hand, when an attention check question in a subtask is correctly answered, all the low-quality unit data in the subtask would be unfortunately accepted; this undesirable situation would occur also to all those 35.03% mixed subtasks and even to those 29.81% subtasks that only contain suspicious unit data.

4.2.3 Guideline on Using FBQC for Suspicious Behavior Detection. For job requesters who have textual emotion recognition or other subjective tasks such as surveys and content quality assessment, they simply need to follow the procedure that we used in our text task experiments to perform either fine-grained or coarse-grained level suspicious behavior detection. In short, they need to: (1) enable the collection of fine-grained behavioral features (e.g., by including our JavaScript library into their task webpages), (2) define some rules based on the heuristics (e.g., expected interactions with certain web elements) that are relevant to their tasks, (3) run simple scripts to inspect if a worker's behavior follows the defined rules, and finally (4) remove submissions with suspicious behaviors (e.g., at either the unit level or the subtask level) to improve the overall quality of the collected annotations or responses. Note that unlike that in quality prediction for objective tasks, no training is needed here in suspicious behavior detection for subjective tasks.

5 UNSUPERVISED WORKER CATEGORIZATION

Prior research studies have demonstrated that categorizing workers into several types and then adopting different quality control strategies accordingly can be very helpful (Section 2). However, automatically identifying different types of workers is challenging. As mentioned in Section 1, we propose to identify different types of workers at the overall task level based on fine-grained behavioral features by using unsupervised clustering techniques.

5.1 Task-level Worker Categorization

In prior studies, researchers labeled workers based on the subtask data [6, 7, 17], thus their proposed worker types represent the subtask level behaviors of workers. Instead of staying at the subtask level, we propose to categorize workers based on time-varying fine-grained behaviors, submission quality, and subtask attributes by considering all subtasks together with their corresponding units completed by a worker in a task. We expect that our task level worker categorization could help capture the overall behavior of a worker for a given type of task, which can facilitate the effective assignment of different tasks to different workers and the identification of malicious workers.

We first performed an exploratory analysis of the data collected from the image and text tasks (described in Section 4) in terms of the task level behavior. We then carefully identified the following four categories of workers at the task level.

Competent Workers. These workers provide high-quality submissions for all their subtasks regardless of the completion time. Specifically, their average IoU value or ratio of non-suspicious unit data in each subtask is higher than 50% workers' average IoUs or ratios in the same subtask. Besides, no suspicious behavior is detected on every unit they completed.

Malicious Workers. These workers seem to be purely money-driven, and attempt to compete each subtask with the least time or effort, i.e., their completion time in each subtask is less than 50% workers' completion time in the same subtask. Meanwhile, in the image task, malicious workers always obtain lower IoU values than other workers, i.e., their average IoU value in a subtask is lower than 50% workers' average IoUs in the same subtask. In the text task, malicious workers always use autofilling tools to complete all units or manually provide lots of invalid responses. For example, a worker has completed 50 dialogues in our text task by always using tools to autofill options. This worker stayed active on each dialogue within about 10 seconds, and annotated all utterances with the same option "neutral". Although the autofilled option "neutral" is reasonable for many utterances, this worker still falls into this category.

Less-competent Workers. These workers appear to have a genuine intent to complete all subtasks successfully by spending sufficient time in each subtask, i.e., their completion time in each subtask is more than 50% workers' completion time in the same subtask. However, majority of their subtask submissions are low-quality for an objective task (i.e., their average IoU value in a subtask is lower than 50% workers' average IoUs in the same subtask), or less reliable for a subjective task (i.e., their ratio of non-suspicious unit data in a subtask is lower than 50% workers' ratios in the same subtask). These workers' behaviors and less-competent performance might be caused by their lack of necessary skills or their unclear understanding of task instructions.

Inconsistent Workers. These workers act like a competent or less-competent worker in some subtasks while act like a malicious worker in others. The varying behaviors and performance of these workers could be related to their time-varying intentions. For example, a worker spent sufficient time and provided high-quality in the first few subtasks, but started to be malicious by using tools to autofill the rest subtasks.

While some worker types such as competent workers seem similar to that proposed in [6], we highlight that our worker types are defined at the task level which is essentially different from

[6]. For example, a worker who completes multiple subtasks will be considered as competent only if the worker is competent in all subtasks. However, in [6] a worker is identified as competent as long as the worker is competent in a subtask. Besides, we introduce inconsistent workers as a new type to reveal a worker's time-varying behaviors across all subtasks within a task. Based on the rubrics above, we manually inspected and identified (or labeled) 85 randomly sampled workers in the image task and 103 randomly sampled workers in the text task as shown in Table 11. To investigate if unsupervised clustering techniques can help properly group workers into four categories, we measure how those manually labeled workers will be distributed after applying clustering techniques with 4 as the number of clusters.

Table 11. Manually Identified or Labeled Types for Sampled Workers

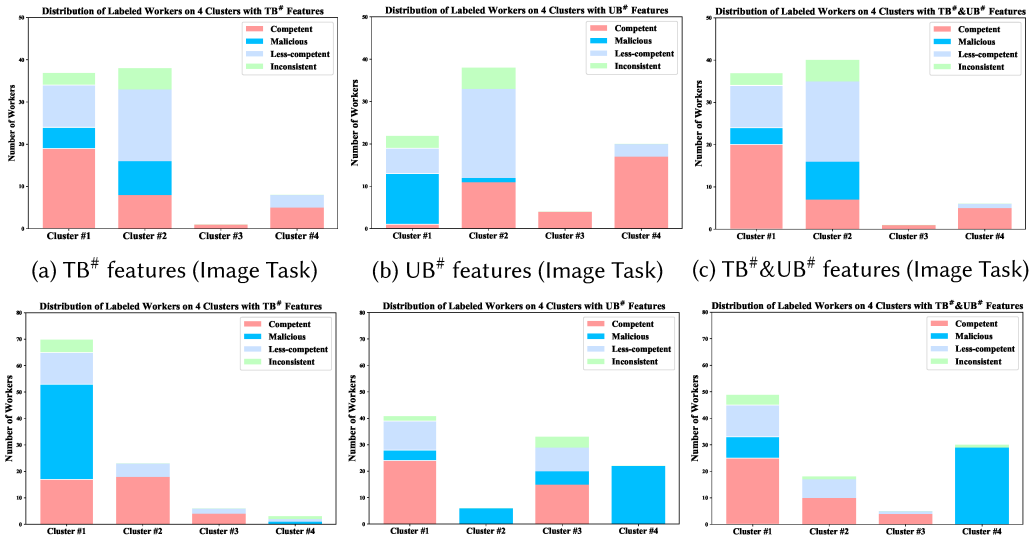
Task (# workers)	Competent	Malicious	Less-competent	Inconsistent
Image (85)	34	13	30	8
Text (103)	39	38	20	6

5.2 Evaluation of Worker Clustering

We apply the popular K-Means clustering technique (n_cluster = 4) on task level data with different types of behavioral features, and evaluate the performance of the clustering method based on the randomly sampled and manually labeled workers listed in Table 11. At the high-level, an effective clustering method should accurately group similar entities together while separating different ones. In particular, we use *purity* [24] as the metric to measure the extent to which each cluster contains a single category. Purity is defined as the percentage of the total number of data points that are correctly grouped; the higher the purity score, the better the performance. Figure 7 shows the distribution of the manually labeled workers on four clusters in six different experiments.

In the image task, majority of the labeled workers are in Cluster #1 and Cluster #2 as shown in Figure 7a when we use $TB^{\#}$ features (described in Section 3.1.2). We have lots of competent workers as well as many less-competent and malicious workers in Cluster #1. We have majority of malicious workers as well as many less-competent and competent workers in Cluster #2. When we cluster workers with $UB^{\#}$ features (described in Section 3.1.2), we have a clearer distinction among malicious workers (Cluster #1), less-competent workers (Cluster #2), and competent workers (Cluster #3 and Cluster #4) as shown in Figure 7b than that in Figure 7a. But when we combine all behavioral features, different types of workers are mixed again as shown in Figure 7c. Meanwhile, the average purity is 0.500 with $TB^{\#}$ features, 0.643 with $UB^{\#}$, and 0.536 with combined $TB^{\#}$ and $UB^{\#}$ features. These results indicate that clustering workers with unit behavioral features performs best in distinguishing different types of workers in the image task.

In the text task, when we cluster workers with $TB^{\#}$ features, majority of labeled workers are located in Cluster #1 as shown in Figure 7d. This cluster mixes malicious workers with other workers, which is especially undesirable. When we cluster workers with $UB^{\#}$ features, majority of malicious workers are in a single cluster (Cluster #4) as shown in Figure 7e. This result validates that clustering with $UB^{\#}$ features performs better than with $TB^{\#}$ features in distinguishing malicious workers from other workers. When we combine all behavioral features, more malicious workers are in Cluster #4 as shown in Figure 7f. Meanwhile, the average purity is 0.578 with $TB^{\#}$ features, 0.650 with $UB^{\#}$ features, and 0.667 with combined $TB^{\#}$ and $UB^{\#}$ features. These results indicate that using unit behavioral features is also helpful in task-level worker clustering in the text task.



(a) TB# features (Image Task) (b) UB# features (Image Task) (c) TB#&UB# features (Image Task)

(d) TB# features (Text Task) (e) UB# features (Text Task) (f) TB#&UB# features (Text Task)

Fig. 7. Distribution of Manually Labeled Workers on Four Clusters in Six Different Experiments (TB - subTask Behavioral Features, UB - Unit Behavioral Features. Here TB# features are statistical features derived from TB features of all subtasks in a task, and UB# features are statistical features derived from UB features of all units in a task as described in Section 3.1.2.)

5.3 Guideline on Using FBQC for Worker Categorization

For job requesters who have either subjective or objective tasks, they simply need to follow the procedure that we used in our image or text task experiments to perform unsupervised worker categorization. Briefly, they need to: (1) enable the collection of fine-grained behavioral features as described before, (2) try some numbers of clusters and apply an unsupervised clustering model (e.g., K-Means) to categorize workers based on the extracted behavioral features, (3) randomly sample some workers from each cluster and manually analyze their overall task level behaviors as well as submission quality to define the worker type for each cluster, (4) improve the overall data quality by applying different strategies to different types of workers such as stopping malicious workers from taking new subtasks and training less-competent workers to improve their skills. Note that a requester could define different worker types based on their specific tasks and observations, or could directly leverage some worker types identified by us and other researchers.

While this guideline is more about a post-hoc analysis, we emphasize that our method could be used to perform real-time or near real-time worker categorization for quality control. To achieve this, job requesters may first recruit a small number of workers for taking some subtasks and collecting their behavioral features along with annotations. They can then apply the above steps (2) and (3) to group those workers into several clusters with the corresponding worker types defined. They can further launch the rest subtasks for either new or existing workers to take. As long as a new worker completes a few subtasks, the collected behavioral features will be used to place the worker into a corresponding cluster. As long as an existing worker completes a new subtask, the collected behavioral features can also be included to refine the cluster placement of this worker. This real-time or near real-time application of unsupervised worker clustering can be helpful for crowdsourcing tasks that contain a large number of subtasks to be completed by many workers, and it can be complementary to the supervised worker type prediction approach taken in [6].

6 DISCUSSION

In this section, we discuss (1) workers' tool usages in our tasks via a survey study, (2) generalizability, deployability, and scalability of the FBQC framework, and (3) limitations and future work.

6.1 Workers' Tool Usage

Based on the analysis of the observed suspicious behaviors (Section 4.2), we suspect that majority of them are related to tool usages. As an additional verification, we conducted a follow-up survey to the workers who performed either of our two tasks. Note that they would not know they were invited to take our survey because they have completed at least one of our two tasks. We only asked general questions about their potential tool usages. We first asked the question "Did you use some tools (e.g., browser extensions, add-ons, scripts) to increase your efficiency while completing tasks on Figure Eight?". If workers answered "Yes", they were further asked to select the features in their tools, such as autofill form fields and auto-refresh a web page.

We obtained responses from 167 workers. In total, 40 workers acknowledged their tool usages, and 11 of them checked "auto-fill form fields (e.g., radio buttons)" as one of the features in their used tools. The rest 127 workers denied using tools in crowdsourcing. By carefully inspecting the recorded behaviors of those 167 workers on our tasks, we found that 7 of the 40 tool-using workers have obvious tool-usage related suspicious behaviors, while no suspicious behaviors are identified for the rest 33 workers. Among the 127 workers who denied using tools, 18 of them have suspicious behaviors in our tasks while the rest 109 workers do not. These results indicate that it is common for workers to use tools to perform crowdsourcing tasks. Fortunately, our proposed framework and fine-grained features can effectively help identify those workers with suspicious behaviors.

6.2 Generalizability, Deployability, and Scalability of the FBQC Framework

As demonstrated in our experiments, the FBQC framework can help support three types of quality control in crowdsourcing: quality prediction, suspicious behavior detection, and worker categorization. It is important to note that this framework can be generally applied to many prominent types of crowdsourcing tasks beyond the visual object detection and textual emotion recognition tasks evaluated in our experiments. This is because many types of tasks naturally contain data at different granularities. Table 12 exemplifies several other important crowdsourcing tasks and their data at different granularities. Those tasks often appear on crowdsourcing platforms and are also often studied in the literature. With their corresponding definitions of the unit data, fine-grained behavioral features can be specifically extracted and analyzed just like what we did for our image and text tasks. For example, in the image segmentation task, fine-grained behavioral features can be extracted from the interactions, including mouse moves, mouse clicks, keypresses, scrolls, etc., that are involved in the generation of the outline of a target object in an image. Meanwhile, without being restricted to the interactions with a specific unit, coarse-grained behavioral features can be extracted from a worker's interactions with all images on a webpage. Similar to Table 3, Table 13 shows the fine-grained features that can be extracted at the unit level for the crowdsourcing tasks listed in Table 12, and similar to Table 4, Table 14 shows the coarse-grained features that can be extracted at the subtask level. Task behavioral features can be derived by aggregating unit and subtask behavioral features at the task level as we explained in Section 3.2.2. Note that more attribute features (UA features in Table 13 and TA features in Table 14) can be explored by job requesters based on the content of a crowdsourcing task. Overall, from the generalizability perspective, it is always feasible to apply our FBQC framework to a task as long as the unit data (and subtask data) concept exists for the task.

Table 12. Examples of Other Important Crowdsourcing Tasks and their Data at Different Granularities.

Crowdsourcing Task	Unit Data	Subtask Data	Task Data
Image Segmentation [11]	The outline of a target object provided by a worker for an image on a webpage.	All outlines of target objects provided by a worker for all images on a webpage.	All outlines of target objects provided by a worker in the entire image segmentation task.
Image Transcription [6]	The content in a text input field provided by a worker for an image on a webpage.	All contents in text input fields provided by a worker for all images on a webpage.	All contents in text input fields provided by a worker in the entire image transcription task.
Text Annotation by Token [23]	The token selected by a worker for a target class in a paragraph on a webpage.	All tokens selected by a worker for target classes in all paragraphs on a webpage.	All tokens selected by a worker for target classes in the entire text annotation task.
Reading Comprehension [34]	The answer provided by a worker to a question in a paragraph on a webpage.	All answers provided by a worker to questions in all paragraphs on a webpage.	All answers provided by a worker to questions in the entire reading comprehension task.
Survey [26, 30]	The response provided by a worker to a question in a survey.	All responses provided by a worker to a section or webpage of questions in a survey.	All response provided by a worker to all questions in the entire survey.
Relevance Judgment [8, 18]	The relevant score provided by a worker for a query-document pair on a webpage.	All relevant scores provided by a worker for all query-document pairs on a webpage.	All relevant scores provided by a worker for all query-document pairs in the entire relevance judgment task.

Table 13. Fine-Grained Features for Each Unit in Other Important Crowdsourcing Tasks

Feature Type	Feature Name	Description
Unit Behavioral (UB) Features	time_on_unit	Time spent on a unit task, i.e., the unit data (Column 2 of Table 12) for certain crowdsourcing task shown in Table 12.
	total_[X]_events	The number of logged events of type X for the unit data (Column 2 of Table 12) where X could be one in {clicks, keypresses, checks, ...} in the crowdsourcing task.
	num_change_annotation	The number of times that a worker changes the annotation (i.e., the created unit data shown in Column 2 of Table 12).
	events_around_annotation	The number of logged events immediately around the annotation action for the unit data, including clicks, keypresses, movements, etc.
	movement_speed_unit	The mean, median, and standard deviation of mouse movement speed within the created unit data.
	speed_around_annotation	The mean, median, and standard deviation of mouse movement speed before/after creating the unit data for certain crowdsourcing task shown in Table 12.
Unit Attribute (UA) Features	unit_attributes	The attributes of a unit (Column 2 of Table 12), e.g., (1) the size/entropy of the created outline and image gradient in the Image Segmentation task, (2) the size and entropy of the given image in the Image Transcription task, (3) the location/length of the token in the Text Annotation by Token task, (4) the number of sentences/words of given paragraphs, the co-occurrence words between paragraphs and a question in the Reading Comprehension task, (5) the number of words of a question in the Survey task, and (6) the number of query words in each document in the Relevance Judgment task.

The FBQC framework is highly deployable by either a job requester or a crowdsourcing platform. For a job requester, our framework can be adopted or implemented in two steps: (1) using JavaScript and some library such as JQuery to log workers' interactions with the task webpages especially at the unit level, and then (2) extracting and analyzing fine-grained behavioral features based on logged events in an automatic or semi-automatic manner. To facilitate the adoption of our framework, we have implemented a JavaScript library that can be easily used by job requesters to obtain fine-grained behavioral features from the webpages of different types of crowdsourcing tasks such as the ones listed in Table 12. A job requester only needs to include our JavaScript library into a corresponding task webpage, from which different types of interactions (such as mouse clicks, keypresses, scrolls, etc.) on individual web elements of different types (such as buttons, check boxes, text fields, etc.) can all be monitored and saved along with the annotations for analysis. For example, to design an image transcription task on the popular Figure Eight crowdsourcing platform

Table 14. Coarse-Grained Features for Each Subtask in Other Important Crowdsourcing Tasks

Feature Type	Feature Name	Description
subTask Behavioral (TB) Features	time_on_subtask	Total time spent on a subtask, i.e., the subtask data (Column 3 of Table 12) for certain crowdsourcing task shown in Table 12.
	total_[X]_events	The number of logged events of type X on a webpage where X could be one in {clicks, keypresses, checks, ...} for certain crowdsourcing task.
	time_on_instruction	Time spent by a worker on reading the task instruction before starting the first unit in a subtask.
	tBeforeInput	Time taken by a worker before creating the first annotation (i.e., the unit data) in a subtask.
subTask Attribute (TA) Features	subtask_attributes	The attributes of a subtask (Column 3 of Table 12), e.g., (1) the size, entropy and image gradients of all given images on a webpage in the Image Segmentation task, (2) the size and entropy of all given images on a webpage in the Image Transcription task, (3) the number of tokens in a paragraph in the Text Annotation by Token task, (4) the number of sentences/words of given paragraphs, and the number of questions in the Reading Comprehension task, (5) the number of questions in the Survey task, and (6) the number of query-document pairs on a webpage in the Relevance Judgment task.

of Appen [1], a job requester can present each image by providing its URL, and create an input text field right after the image using "cml:text"[5], which is a helper tag on Appen used for rendering a single-line text field. To apply our framework, the job requester can simply include our JavaScript library in the "Javascript" field on the design webpage, specify the address of a server for trace collection, launch the task, and analyze the behavioral and attribute features (extracted from both images and text fields) for quality control.

Our framework is also highly deployable by crowdsourcing platforms as they can directly add the JavaScript code to their task webpage templates or generation engines. They can provide FBQC as one of the optional quality control approaches, thus job requesters can directly utilize it by simply enabling the option in the quality control settings. When implementing or enabling the framework, we argue that it should be done with transparency to protect workers' rights. In our study, we have reminded workers in the consent form that we will "examine how you complete our text or image related tasks". In the long run, explicitly informing workers about the existence of behavior-based quality control may encourage them to more responsibly perform the tasks, and may also help deter malicious workers.

The good generalizability and deployability of the FBQC framework imply that this framework is scalable from the perspectives of being applicable to different types of tasks and being adoptable by either a job requester or a crowdsourcing platform. Another perspective of scalability is related to the concern on the consumption of a worker's computing resources. It is worth noting that the resource consumption overhead incurred by our framework to a worker's web browser and computer is minimal if not negligible. We measured the page load time[6], memory usage, and CPU usage on our image and text subtask webpages using a laptop with a 2.9 GHz Intel i5 processor, 8GB of RAM, and a Google Chrome browser. Comparing between using and without using our JavaScript code to collect behavioral traces, our framework introduced an average of 72 ms page load overhead, 2.92 MB memory overhead, and 0.995% CPU usage overhead on both tasks. Qualitatively, we cannot perceive any page load delay or slowdown of our web browser; meanwhile, workers on the Appen platform can submit comments to job requesters, but we did not receive any complaint about the page load time or resource consumption on our tasks. Intuitively, our JavaScript code mainly reads the attributes of HTML *<input>* elements on a webpage as described in Section 3.1; its execution is far less intense than that of a typical News or Map webpage.

[5]https://success.appen.com/hc/en-us/articles/202815199-Form-Element-Single-Line-Text-Input

[6]The page load time is measured by the time difference between the browser's *navigationStart* and *loadEventEnd* events on a webpage.

6.3 Limitations and Future Work

In our experiments, we decided the payment rate for each task based on the time estimations gathered from our pilot study and the minimum wage of 7.25 USD/hr in the United States [10] as described in Section 3.2. Our post-hoc analysis on the subtask completion time showed that majority of subtasks are paid above the minimum wage. However, in future tasks, we will adopt more flexible payment methods such as setting the payment rate based on the time spent by workers [25, 35], or we will simply pay more than the basic estimations to leave enough margin for ensuring a fairer payment rate to workers.

As shown in Section 4.1, our supervised learning models achieve relatively high accuracy on quality prediction for our image task at different levels. We acknowledge that the prediction accuracy would vary for different types of objective tasks, so building models and evaluating their performance for other types of tasks could be helpful in the future. Meanwhile, we acknowledged in Section 4.1.3 that a training dataset with the ground-truth labels must be constructed for our quality prediction models. One potential future work for addressing the training dataset construction problem is to leverage labeled samples from other related tasks and use transfer learning techniques.

As described in Section 4.2, our method achieves a 90% accuracy in detecting suspicious behaviors in the text task. However, we still have some false positives and false negatives. Improving rules to reduce those false cases could be one important future work. In addition, although our experimental analysis reveals that the majority of the detected suspicious behaviors are related to tool usages, we are aware that our rules may not work for detecting suspicious behaviors that are purely related to manual submissions. Namely, malicious workers could escape our detection by manually providing invalid annotations, e.g., choosing the options at random. Note that this is a common limitation of behavior-based quality control mechanisms. In Section 4.2.2, we compared our suspicious behavior detection mechanism with the widely used attention check mechanism more from the perspective of detecting tool usage related suspicious behaviors. Here, even the attention check mechanism may fail to detect manual submission related suspicious behaviors. This is because when malicious workers manually check each unit, it would be easy for them to identify attention check questions and provide correct answers. Detecting manual submission related suspicious behaviors is still a challenging future work. However, it is common for workers to use tools to perform crowdsourcing tasks as analyzed in Section 6.1. Therefore, more generally and in the future, to achieve better quality control and to provide more support to workers as well, thorough research on workers' tool usages is important and necessary.

In addition to predicting data quality and detecting suspicious behaviors using fine-grained behavioral features, another contribution of our work is task-level worker categorization (described in Section 5). However, our task-level worker categorization concerns worker behaviors in a specific crowdsourcing task, thus may have limitations in helping understand workers' comprehensive characteristics, such as how a worker performs differently or similarly across different types of tasks. Although we have 47 workers completed both the text and image tasks, the number of workers and number of tasks are not big enough for us to derive some conclusive results at this moment regarding their behaviors across different tasks. An in-depth understanding of how a worker performs in different types of tasks would facilitate the worker selection and task assignment processes by making them more intelligent. Therefore, exploring and investigating workers' behaviors across multiple and different types of crowdsourcing tasks would be another promising future work.

7 CONCLUSION AND FUTURE WORK

Behavioral features of workers have been demonstrated to be highly correlated with data quality, thus have become increasingly important for quality control in crowdsourcing. In this paper, we

proposed to explore the feasibility and benefits of using fine-grained behavioral features for quality control at the fine-grained level and also at higher levels. We designed and implemented the FBQC framework that specifically extracts fine-grained behavioral features to provide three quality control mechanisms: (1) quality prediction for objective tasks, (2) suspicious behavior detection for subjective tasks, and (3) unsupervised worker categorization. Using the FBQC framework, we conducted two real-world crowdsourcing experiments and demonstrated that using fine-grained behavioral features are feasible and beneficial in all three quality control mechanisms. We also discussed our study from multiple perspectives including workers' tool usages, generalizability, deployability, scalability, limitations, and future work. Our work provides clues and implications for helping job requesters or crowdsourcing platforms to further achieve better quality control.

ACKNOWLEDGMENTS

We thank anonymous reviewers and ACs for their valuable suggestions. This research was supported by the NSF grant OIA-1936968.

REFERENCES

[1] Appen 2021. Confidence to Deploy AI with World-Class Training Data. https://appen.com/.
[2] Natã M. Barbosa and Monchu Chen. 2019. Rehumanized Crowdsourcing: A Labeling Framework Addressing Bias and Ethics in Machine Learning. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*.
[3] Alessandro Checco, Jo Bates, and Gianluca Demartini. 2020. Adversarial Attacks on Crowdsourcing Quality Control. *Journal of Artificial Intelligence Research* (2020).
[4] Sheng-Yeh Chen, Chao-Chun Hsu, Chuan-Chun Kuo, Lun-Wei Ku, et al. 2018. Emotionlines: An emotion corpus of multi-party conversations. *arXiv preprint arXiv:1802.08379* (2018).
[5] Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* (1979).
[6] Ujwal Gadiraju, Gianluca Demartini, Ricardo Kawase, and Stefan Dietze. 2019. Crowd anatomy beyond the good and bad: Behavioral traces for crowd worker modeling and pre-selection. *Computer Supported Cooperative Work (CSCW)* (2019).
[7] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*.
[8] Tanya Goyal, Tyler McDonnell, Mucahid Kutlu, Tamer Elsayed, and Matthew Lease. 2018. Your Behavior Signals Your Reliability: Modeling Crowd Behavioral Traces to Ensure Quality Relevance Annotations. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.
[9] Shuguang Han, Peng Dai, Praveen Paritosh, and David Huynh. 2016. Crowdsourcing human annotation on web page structure: Infrastructure design and behavior-based quality control. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2016).
[10] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. 2018. A data-driven analysis of workers' earnings on Amazon Mechanical Turk. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*.
[11] Eric Heim, Alexander Seitel, Jonas Andrulis, Fabian Isensee, Christian Stock, Tobias Ross, and Lena Maier-Hein. 2017. Clickstream analysis for crowd-based object segmentation with confidence. *IEEE transactions on pattern analysis and machine intelligence* (2017).
[12] Danula Hettiachchi, Niels van Berkel, Vassilis Kostakos, and Jorge Goncalves. 2020. CrowdCog: A Cognitive skill based system for heterogeneous task assignment and recommendation in crowdsourcing. *Proceedings of the ACM on Human-Computer Interaction* CSCW2 (2020).
[13] Matthias Hirth, Sven Scheuring, Tobias Hoßfeld, Christian Schwartz, and Phuoc Tran-Gia. 2014. Predicting result quality in crowdsourcing using application layer monitoring. In *Proceedings of IEEE International Conference on Communications and Electronics (ICCE)*.
[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).
[15] Ayush Jain, Akash Das Sarma, Aditya Parameswaran, and Jennifer Widom. 2017. Understanding workers, developing effective tasks, and enhancing marketplace dynamics: a study of a large crowdsourcing marketplace. *Proceedings of the VLDB Endowment* (2017).
[16] jQuery 2021. jQuery. https://jquery.com/.

[17] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. 2011. Worker types and personality traits in crowdsourcing relevance labels. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.
[18] Gabriella Kazai and Imed Zitouni. 2016. Quality management in crowdsourcing using gold judges behavior. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.
[19] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV* (2020).
[20] Walter S Lasecki, Jaime Teevan, and Ece Kamar. 2014. Information extraction and manipulation threats in crowd-powered systems. In *Proceedings of the ACM conference on Computer Supported Cooperative Work & Social Computing (CSCW)*.
[21] Hongwei Li and Bin Yu. 2014. Error rate bounds and iterative weighted majority voting for crowdsourcing. *arXiv preprint arXiv:1411.4086* (2014).
[22] Ioanna Lykourentzou, Angeliki Antoniou, Yannick Naudet, and Steven P Dow. 2016. Personality matters: Balancing for personality types leads to better outcomes for crowd teams. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW)*.
[23] Chris Madge, Juntao Yu, Jon Chamberlain, Udo Kruschwitz, Silviu Paun, and Massimo Poesio. 2019. Crowdsourcing and aggregating nested markable annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
[24] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*. Cambridge university press.
[25] Andrew Mao, Ece Kamar, Yiling Chen, Eric Horvitz, Megan Schwamb, Chris Lintott, and Arfon Smith. 2013. Volunteering versus work for pay: Incentives and tradeoffs in crowdsourcing. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.
[26] Asako Miura and Tetsuro Kobayashi. 2016. Survey satisficing inflates stereotypical responses in online experiment: The case of immigration study. *Frontiers in psychology* (2016).
[27] Ricky KP Mok, Rocky KC Chang, and Weichao Li. 2016. Detecting low-quality workers in QoE crowdtesting: A worker behavior-based approach. *IEEE Transactions on Multimedia* (2016).
[28] MTurk 2018. Amazon Mechanical Turk. https://www.mturk.com.
[29] Daniel M Oppenheimer, Tom Meyvis, and Nicolas Davidenko. 2009. Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of experimental social psychology* (2009).
[30] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. 2010. Running experiments on amazon mechanical turk. *udgment and Decision Making* (2010).
[31] Weiping Pei, Arthur Mayer, Kaylynn Tu, and Chuan Yue. 2020. Attention please: Your attention check questions in survey studies can be automatically answered. In *Proceedings of The Web Conference 2020*.
[32] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 527–536.
[33] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
[34] Jeffrey M Rzeszotarski and Aniket Kittur. 2011. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proceedings of the ACM Symposium on User Interface Software and Technology*.
[35] Susumu Saito, Chun-Wei Chiang, Saiph Savage, Teppei Nakano, Tetsunori Kobayashi, and Jeffrey P Bigham. 2019. TurkScanner: Predicting the hourly wage of microtasks. In *Proceedings of The World Wide Web Conference (WWW)*.
[36] Rachel N Simons, Danna Gurari, and Kenneth R Fleischmann. 2020. " I Hope This Is Helpful" Understanding Crowdworkers' Challenges and Motivations for an Image Description Task. *Proceedings of the ACM on Human-Computer Interaction* CSCW2 (2020).
[37] Yu Suzuki, Yoshitaka Matsuda, and Satoshi Nakamura. 2019. Additional operations of simple HITs on microtask crowdsourcing for worker quality prediction. *Journal of Information Processing* (2019).
[38] Jeroen BP Vuurens and Arjen P De Vries. 2012. Obtaining high-quality relevance judgments using crowdsourcing. *IEEE Internet Computing* (2012).
[39] Haijun Zhai, Todd Lingren, Louise Deleger, Qi Li, Megan Kaiser, Laura Stoutenborough, and Imre Solti. 2013. Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing. *Journal of medical Internet research* (2013).

Received January 2021; revised April 2021; revised July 2021; accepted July 2021