Leveraging Peer Communication to Enhance Crowdsourcing

Wei Tang
Washington University in St. Louis
w.tang@wustl.edu

Chien-Ju Ho
Washington University in St. Louis
chienju.ho@wustl.edu

Ming Yin
Purdue University
mingyin@purdue.edu

ABSTRACT

Crowdsourcing has become a popular tool for large-scale data collection where it is often assumed that crowd workers complete the work *independently*. In this paper, we relax such independence property and explore the usage of *peer communication*—a kind of direct interactions between workers—in crowdsourcing. In particular, in the crowdsourcing setting with peer communication, a *pair* of workers are asked to complete the same task together by first generating their initial answers to the task independently and then freely discussing the task with each other and updating their answers after the discussion. We first experimentally examine the effects of peer communication on individual microtasks. Our results conducted on three types of tasks consistently suggest that work quality is significantly improved in tasks with peer communication compared to tasks where workers complete the work independently. We next explore how to utilize peer communication to optimize the requester's total utility while taking into account higher data correlation and higher cost introduced by peer communication. In particular, we model the requester's online decision problem of whether and when to use peer communication in crowdsourcing as a constrained Markov decision process which maximizes the requester's total utility under budget constraints. Our proposed approach is empirically shown to bring higher total utility compared to baseline approaches.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in HCI; Computer supported cooperative work**.

KEYWORDS

crowdsourcing; peer communication; collaboration

ACM Reference Format:
Wei Tang, Chien-Ju Ho, and Ming Yin. 2019. Leveraging Peer Communication to Enhance Crowdsourcing. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3308558.3313554

1 INTRODUCTION

Crowdsourcing has gained increasing popularity recently as a scalable data collection tool for various purposes, such as obtaining labeled data for training machine learning algorithms and getting high-quality yet cheap transcriptions for audio files. On a typical crowdsourcing platform like Amazon Mechanical Turk (MTurk), task requesters can post small jobs (e.g., image labeling or audio transcription tasks) as "microtasks" along with the specified payment for completing each task. Workers then can browse all available tasks on the platform and decide which ones to work on. Crowd workers are often assumed to complete tasks *independently*, and a substantial amount of crowdsourcing research has been focused on how to make better use of the independent workers. For example, a rich body of research has explored how to aggregate independent contributions from multiple workers by inferring task difficulties, worker skills, and correct answers simultaneously [10, 41, 49, 53]. Moreover, given a limited budget, researchers have further examined how to intelligently decide the number of independent workers needed for each task at the first place [8, 21, 34].

However, the validity and value of this independence assumption in crowdsourcing has been challenged recently. Through a combination of ethnographic and experimental methodologies, researchers have found that crowd workers, in fact, communicate and collaborate with each other through both online forums (like TurkerNation[1] and MTurkCrowd[2]) and one-on-one channels [19, 20, 25, 37, 44, 51]. Different from such collaboration which is organically arisen within the crowd and mostly about exchanging meta-level information related to crowd work (e.g., how to find well-paid tasks), an increasing number of studies in the human-computer interaction community have started to design certain level of interactions between workers in their actual work, which is shown to improve crowdsourcing outcomes in many cases. For example, various *workflows* are developed to coordinate workers to work on *different* subtasks and interact with each other through the pre-defined input-output handoffs [4, 9, 30–32, 35, 40, 42], which enable the crowd to jointly complete complex tasks.

More recently, worker interactions are further introduced between workers of the *same* task: Drapeau et al. [18] and Chang et al. [7] showed that in image/text labeling tasks, workers can improve their labeling accuracy when *indirect* interactions—in the form of showing each worker the alternative answer and associated justification produced by another worker who works on the same task—are enabled, and Schaekermann et al. [45] observed that in text classification tasks, worker performance increases when they can debate their answers through *direct, real-time* deliberation with one another. While these research show the promise of an alternative way to structure crowd work that leads to higher performance, they also raise a number of open questions.

First, on the "micro" level, it is important to empirically examine whether adding interactions between workers working on the same task leads to an increase in work quality for *individual* tasks of *different* types, especially for those tasks with a large number of possible answers (rather than just a few options as in image labeling and text classification tasks). Indeed, when the number of possible

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313554

[1] http://turkernation.com/
[2] https://www.mturkcrowd.com/

answers in a task becomes large, workers may hardly agree with each other so it is unclear whether interactions between them would be meaningful and effective. It is also impractical for workers to argue against all alternative answers during their interactions, which may imply the need for new formats of interactions beyond providing justification and argumentation.

Furthermore, from a more "macro" point of view, requesters typically have a *large batch* of tasks at hand and need to solicit answers from multiple workers for each task. Their goal is to optimize their *overall* utility such as maximizing the quality obtained across *all* the tasks under a fixed budget. Yet, compared to independent work, allowing worker interactions in a task can bring up not only work quality improvement in that task, but also higher cost and higher correlation in workers' answers. Thus, for requesters to make better use of worker interactions, a critical problem to address is that given a limited budget, *whether* and *when* should worker interactions be used in each task, such that after combining the possibly correlated answers together for each task, the work quality for the entire batch of tasks is maximized when the budget is exhausted.

In this paper, we attempt to answer these two questions. In particular, inspired by the concept of peer instruction in education [12], we focus on studying a specific format of worker interactions that we refer to as *peer communication*—a *pair* of workers working on the same task are asked to first provide an independent answer each, then freely discuss the task with each other, and finally provide an updated answer, again independently, after the discussion. Compared to worker interaction formats used in the early research (e.g., justification and argumentation), we consider peer communication as a kind of direct and synchronous interaction that can be generalized to different types of tasks more easily. Our goal is to better understand not only whether and how peer communication would affect the outcome of crowd work for various types of tasks, but also how requesters can use algorithmic approaches to better utilize the potential benefits brought up by peer communication.

To understand the effects of peer communication on crowd work, we design and conduct randomized experiments with three different types of tasks: image labeling, optical character recognition, and audio transcription. For all types of tasks in our experiments, regardless of how large the number of possible answers in the task is, we have consistently observed an increase in work quality when workers can talk with their peers in the task compared to workers who work independently. Yet, we do not observe any spillover effects of such quality improvement when workers who have engaged in peer communication work on similar tasks again independently.

Moreover, to examine how peer communication can be better utilized, we propose an algorithmic framework to help requesters make *online decisions* on whether and when to use peer communication for each task in their batch, with the goal of maximizing the overall work quality produced in all tasks given a budget constraint. One of the key challenges here is how to infer the correct answer for a task given multiple answers solicited from workers, where some of them may be produced following the peer communication procedure and thus may be correlated. To this end, we introduce the notions of *meta-workers* and *meta-labels* to describe a pair of workers who have engaged in a task with peer communication and the pair of answers produced by them. Such notions enable us to characterize the possible correlation in data, which further allow us to solve the requester's online decision-making problem by modeling it as a constrained Markov decision process.

We evaluate the effectiveness of the proposed algorithmic approach on real data collected through our experimental study. Results show that using our approach to decide the usage of peer communication in tasks, the requester can achieve higher overall quality across all her tasks when the budget is exhausted, compared to when baseline approaches are adopted where peer communication is always used in all tasks or never used in any of the tasks, or when correlation in data is not explicitly considered. In addition, through two sets of simulated experiments, we further examine how the proposed algorithmic approach performs in various scenarios when the differences in work quality and cost between hiring pairs of communicating workers and hiring independent workers vary, and when answers produced by pairs of communicating workers are correlated to different extent.

In summary, we make the following contributions:

- We introduce peer communication, a general mechanism adapted from the concept of peer instruction in education for including worker interactions in crowd work.
- We empirically show that on different types of tasks, compared to independent work, peer communication consistently leads to a 32%–47% improvement in work quality for individual tasks.
- We propose an algorithmic framework to help requesters dynamically decide whether and when to use peer communication for each task in their batch, so as to maximize the overall quality obtained across all tasks given a budget constraint.
- Through evaluations on both real data from crowd workers and synthetic data, we demonstrate that compared to baseline approaches, using our proposed algorithm to determine the deployment of peer communication leads to higher requester utility.

2 RELATED WORK

Our work joins a long line of research on improving the quality of crowd work. In traditional settings where it is assumed that workers independently complete tasks, various methods have been proposed to address this problem, including post-hoc aggregation of workers' answers [10, 14, 15, 26, 41, 49, 53], designing effective extrinsic incentives [23, 24, 38, 50] and intrinsic motivation [33, 43, 47], appropriate assignment of tasks to workers [22, 27, 28], etc.

We explore how to improve the quality of crowd work from a different angle, that is, by adding interactions between workers. Researchers have previously designed *workflows* for complex tasks to allow workers to work on different subtasks while indirectly interacting with one another through the pre-defined input-output handoffs [4, 9, 30–32, 35, 40, 42]. Different from these workflow-based approaches, we consider the addition of interactions between workers of the *same* task. A few previous studies [7, 18, 45] have showed that enabling interactions between workers working on the same task, in the form of asking workers to provide justification for their answers, can lead to improvement in work quality, but these studies only test this idea on classification tasks. We aim to extend this idea to a wider range of tasks, especially for tasks with a large number of possible answers.

In this paper, we study a specific format of interactions, *peer communication*, which is adapted from "peer instruction" [12] and "think-pair-share" strategies [36] in the educational settings. There is a rich literature in the collaborative learning community suggesting that asking students to discuss conceptual questions with other students after they independently answer the questions leads to higher levels of understanding and post-test performance [11, 16, 48]. We thus design the peer communication procedure as first asking a pair of workers to provide an independent answer each, then allowing them to *freely* discuss the task with each other, and finally independently update their answers. While evidence in the collaborative learning community and results for adding argumentation in classification tasks seem to indicate peer communication would lead to higher work quality, other studies showed that allowing workers to chat during work doesn't change work quality [52]. Thus, it is necessary to re-examine the impact of peer communication on the quality of crowd work, if any. In addition, as prior research on inter-task effects [1, 6, 39] suggests that when working on a sequence of tasks, workers' responses for later tasks could be influenced by the earlier tasks, we further examine whether peer communication brings any "spillover" effect on work quality. That is, whether workers produce higher independent work quality after engaging in similar tasks with peer communication.

Besides empirically showing the benefits of peer communication on work quality, we further provide an algorithmic framework for helping requesters better utilize such benefits. Early work has explored how indirect interactions between workers of different tasks that are embedded in certain workflows can be algorithmically controlled in order to maximize requester utility [13]. In contrast, the purpose of our algorithmic framework is to dynamically decide whether and when to deploy peer communication between workers of the *same* task for each task in requesters' batch to maximize their utility. Our framework is built on top of the work by Chen et al. [8], in which they used a Markov decision process to sequentially decide which task in requesters' batch needs an additional worker to work on given a budget constraint. However, our framework has a few key differences: First, in addition to choose which task needs further work, we also decide on how to design that piece of work—hiring one independent worker or two communicating workers? Second, when making inference for each task, we need to consider the possible correlation in the answers for this task. Finally, since peer communication and independent work incurs *different cost*, this decision-making problem does *not* degenerate into a finite-horizon Markov decision process. Thus, we explicitly model the problem as a *constrained* Markov decision process.

3 EXAMINING PEER COMMUNICATION VIA REAL-WORLD EXPERIMENTS

In this section, we first present our experimental study, in which we carefully examine the effects of introducing peer communication between pairs of workers on the quality produced in individual tasks through a set of randomized experiments conducted on Amazon's Mechanical Turk (MTurk). In particular, we ask:

- **Question 1 (Q1)**: Do workers produce higher work quality in tasks with peer communication compared to that in tasks where workers work independently?

Previous studies on the effects of adding worker interactions in image and text classification tasks [7, 18, 45] seem to imply a positive answer for Q1. Compared to these studies, our study has two key differences that warrant a re-examination of Q1: (1) the main format of interaction in peer communication is a *synchronous, free-form chat* rather than required justification and argumentation; (2) we consider different types of tasks beyond classification, especially tasks with a large number of possible answers so workers can hardly agree with each other or argue against all alternative answers. Moreover, we are also interested in examining whether there is any "spillover" effects of the impact of peer communication on work quality. Specifically:

- **Question 2 (Q2)**: Do workers produce higher independent work quality after engaging in similar tasks with peer communication, compared to workers who always complete tasks on their own?

Both positive and negative answers might be possible for Q2: On the one hand, if communication between workers in tasks allow them to resolve misconception about the tasks or learn useful problem-solving strategies from each other, we might expect a positive answer; on the other hand, if the benefits of peer communication are mostly due to workers being able to exchange their confidence levels on a task and eventually converge to the more confident answer [2], the answer for Q2 would likely be negative.

3.1 Independent Tasks vs. Discussion Tasks

In our experiments, we considered two ways to structure the tasks:

- **Independent tasks** (tasks without peer communication). In an independent task, workers are instructed to complete the task on their own.
- **Discussion tasks** (tasks with peer communication). In discussion tasks, we designed a procedure which guides workers to communicate with each other and complete the task together. Specifically, each worker is paired with another "co-worker" on a discussion task. Both workers in the pair are first asked to work on the task and submit their answers independently. Then, the pair enters a chat room, where they can see each other's independent answer. Workers are instructed to freely discuss the task with their co-workers for two minutes; for example, they can explain to each other why they believe their answers are correct. After the discussion, both workers get the opportunity to independently update and submit their final answers.

3.2 Experimental Treatments

The most straight-forward experimental design would include two treatments, where workers in one treatment are asked to work on a sequence of independent tasks while workers in the other treatment complete a sequence of discussion tasks. However, if we adopt such a design, the different nature of independent and discussion tasks (e.g., discussion tasks require more time and effort from workers but can be more interesting to workers) implies the possibility of observing severe self-selection biases in the experiments (i.e., workers may self-select into the treatment that they can complete tasks faster or find more enjoyable).

To overcome the drawback of this simple design, we design our experiments in a way that each treatment consists of the same number of independent tasks *and* discussion tasks, so neither treatment appears to be obviously more time-consuming or enjoyable. Figure 1 illustrates the two treatments used in our experiments. In

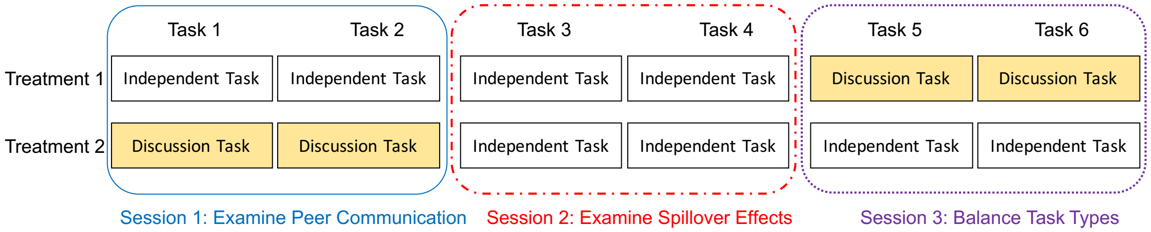

Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Treatment 1
Treatment 2
Independent Task
Discussion Task
Session 1: Examine Peer Communication
Session 2: Examine Spillover Effects
Session 3: Balance Task Types

Figure 1: The two experimental treatments. This design enables us to examine whether peer communication improves the quality of crowd work (by comparing work quality in Session 1) and if so, does the improvement spill over to the following independent tasks (by comparing work quality in Session 2), while not creating significant differences between the two treatments (by adding Session 3 to make the two treatments containing equal number of independent and discussion tasks).

particular, we bundle 6 tasks in each HIT (i.e., Human Intelligence Task on MTurk). When a worker accepted our HIT, she was told that there are 4 independent tasks and 2 discussion tasks in the HIT. There are two treatments in our experiments:

- *Treatment 1*: Workers are asked to complete 4 independent tasks followed by 2 discussion tasks.
- *Treatment 2*: Workers are asked to complete 2 discussion tasks followed by 4 independent tasks.

Importantly, we did *not* tell workers the ordering of the 6 tasks, which helps us to minimize the self-selection biases as the two treatments look the same to workers. We refer to the first, middle, and last two tasks in the sequence as Session 1, 2, 3 of the HIT, respectively. Thus, we can answer Q1 by comparing the work quality produced in Session 1 between the two treatments, while a comparison of work quality in Session 2 between the two treatments would allow us to answer Q2. Finally, Session 3 is used for balancing the number of independent and discussion tasks in each HIT.

3.3 Experimental Tasks

We conducted our experiments on three types of tasks:

- **Image labeling.** In each task, the worker is asked to identify whether the dog shown in an image is a Siberian Husky or a Malamute. Dog images we use are collected from the Stanford Dogs dataset [29].
- **Optical character recognition (OCR).** In each task, the worker is asked to transcribe a vehicle's license plate numbers from photos. The photos are taken from the dataset provided by Shah and Zhou [46].
- **Audio transcription.** In each task, the worker is asked to transcribe an audio clip which contains about 5 seconds of speech. The audio clips are collected from VoxForge[3].

We decided to conduct our experiments on these three types of tasks for two main reasons: First, these tasks are all very common types of tasks on crowdsourcing platforms [17], so experimenting with them would allow us to better understand the effects of peer communication on typical kind of crowd work. Second, in terms of the number of possible answers, these tasks span a wide spectrum from two (image labeling) to infinitely many (audio transcription), enabling us to both confirm the effects of peer communication in tasks with just a few possible answers and explore its effects in tasks with many possible answers. As a final note, tasks we bundled in the same HIT had a certain degree of similarity[4], hence a spillover effect of peer communication on work quality is not impossible as knowledge/strategy that workers may learn in one task can potentially be transferred to another task.

3.4 Experimental Procedure

Enabling synchronous work among crowd workers is quite challenging, as discussed in previous research on real-time crowdsourcing [3, 5]. We address this challenge by dynamically matching pairs of workers and sending them to simultaneously start working on the same sequence of tasks. In particular, when each worker arrived at our HIT, we first checked whether there was another worker in our HIT who didn't have a co-worker yet—if yes, she would be matched to that worker and assigned to the same treatment and task sequence as that worker, and the pair then started working on their sequence of tasks together. Otherwise, the worker would be *randomly* assigned to one of the two treatments as well as a *random* sequence of tasks, and she would be asked to wait for another co-worker to join the HIT for a maximum of 3 minutes. In the case where no other workers arrived at our HIT within 3 minutes, we asked the worker to decide whether she was willing to complete all tasks in the HIT on her own (and we dropped the data for the analysis but still paid her accordingly) or get a 5-cent bonus to keep waiting for another 3 minutes.

We provided a base payment of 60 cents for all our HITs. In addition to the base payments, workers were provided with the opportunity to earn performance-based bonuses, that is, workers can earn a bonus of 10 cents in a task if the final answer they submit for that task is correct. Our experiment HITs were open to U.S. workers only, and each worker was only allowed to take one HIT for each type of tasks.

3.5 Experimental Results

In total, we have 388, 382, and 250 workers who successfully formed pairs and completed the image labeling, OCR, and audio transcription tasks in our experiments, respectively. We then answer Questions 1 and 2 separately for each type of task by analyzing experimental data from Session 1 and 2 in the HIT, respectively.[5]

[3] http://www.voxforge.org

[4] For example, image labeling tasks are all about the key concept of distinguishing Siberian Husky from Malamute, OCR tasks have similar image quality, and audio transcription tasks contain similar accents.

[5] On a side note, analyzing the data collected in Session 3 leads to conclusions that are consistent with our findings reported below, and including such data only strengthens our results. However, since we have decided not to use it in the experiment design phase, we do not include the data in the analysis. The reason of the decision is that

3.5.1 Work Quality Metrics. For all three types of tasks, we evaluate the work quality using the notion of *error*. In the image labeling task, we define error as the binary classification error—the error is 0 for correct labels and 1 for incorrect labels. For OCR and audio transcription tasks, we define error as the edit distance between the worker's answer and the correct answer, divided by the number of characters in the correct answer. Naturally, for all tasks, a lower rate of error implies higher work quality.

3.5.2 Q1: Peer Communication Improves Work Quality. In Figure 2, We plot the average error rate for workers' final answers in the first two tasks (i.e., Session 1) of Treatment 1 and 2 using white and black bars, respectively. Visually, it is clear that for all three types of tasks, the work quality is higher in discussion tasks (i.e., Session 1 of Treatment 2 HITs) when workers are able to communicate with others about the work, compared to that in independent tasks (i.e., Session 1 of Treatment 1 HITs) where workers need to complete the work on their own. Indeed, we observe a substantial 37%, 32%, and 47% deduction in the average error rate for image labeling, OCR, and audio transcription tasks when peer communication is enabled. We further conduct two-sample t-tests to check whether these changes are statistically significant, and p-values are 2.42×10^{-4}, 5.02×10^{-3}, and 1.95×10^{-11} respectively, suggesting that introducing peer communication in crowd work can significantly improve the work quality produced for various types of tasks.

We then look into the chat logs to gain some insights on how and what workers have communicated with each other during the discussion. On average, the length of the discussions in image labeling, OCR and audio transcription tasks are 4.2, 5.1 and 5.4 turns[6], yet the amount of discussion is not correlated to the quality of worker's final answers after discussion. Furthermore, by looking into the content of discussions, we find several types of information workers are exchanging during their communication:

- *Providing Justification*: e.g., "triangle ears that stand erect are traits of a Siberian Husky" (image labeling)
- *Communicating Confidence*: e.g., "I'm pretty sure about UR to start, but not very sure after that" (OCR); "I had no idea what the last word was" (audio transcription)
- *Exchanging Strategy*: e.g., "If you can zoom in on it you will see what I mean" (OCR); "He pronounces 'was' with a v-sound instead of the w-sound" (audio transcription)
- *Expressing Agreement*: e.g., "I agree"; "Listening to it again, I think you are right" (audio transcription)
- *Collaborative Work*: the pair of workers work together to solve the task, e.g., guessing a digit on the car plate for the OCR task that neither worker can recognize independently

Interestingly, as an anecdotal observation, we notice that in image labeling tasks the majority of workers tend to provide justifications for their answers. In OCR and audio transcription tasks, instead of "defending" their own answers, many more workers choose to team up with their co-workers to solve the task together.

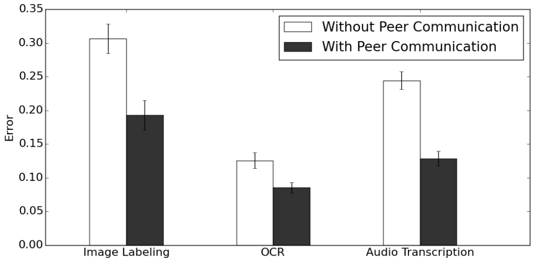


Figure 2: Comparisons of work quality produced in tasks with or without peer communication. Error bars indicate the mean ± one standard error.

3.5.3 Q2: There are no spillover effects. We now move on to Q2: Compared to workers who always complete tasks independently, do workers who have participated in tasks with peer communication continue to produce work of higher quality in future tasks of the same type, even if they need to complete those tasks on their own? To answer this question, we compare the work quality produced in Session 2 (i.e., the middle two independent tasks) of the two treatments for all three types of tasks. For image labeling, OCR, and audio transcription tasks, the average error rates for Session 2 in Treatment 1 (workers never engage in peer communication) are 0.324, 0.175, and 0.209 respectively, while the average error rates for Session 2 in Treatment 2 (workers have previously engaged in peer communication) are 0.334, 0.168, and 0.244. Thus, we do not observe any spillover for the effects of peer communication on work quality, that is, the quality improvement brought up by peer communication does not carry on to future independent work.

3.5.4 Discussions. Results of our experimental study suggest that peer communication improves the quality of crowd work for various types of tasks, even when the number of possible answers in the task is very large, yet such effect does not spill over to later independent work. Cautions should be used when generalizing these results to substantially different contexts, such as when workers can interact with each other for an extended period of time rather than just 2 minutes, when the tasks are significantly more complex or more subjective, or when workers engage in peer communication for a longer sequence of tasks. It is, thus, an important future direction to obtain an thorough understanding on how tuning various parameters of the design space (e.g., length of interactions, complexity/subjectivity of tasks) would change the effects of peer communication.

4 AN ALGORITHMIC FRAMEWORK FOR UTILIZING PEER COMMUNICATION

Our experimental study focuses on understanding the impact of peer communication on *individual tasks*. Now, we turn to our next question, that is, for a requester who has a *large batch* of tasks, how can he better utilize peer communication to improve the *overall utility* that he can obtain across all the tasks? In particular, we address the following research question: *Given a budget and a batch of tasks to complete, whether and when should a requester deploy peer communication in each task to maximize his total utility?*

To answer this question, there are two main challenges. First, when peer communication is deployed, workers communicate with

workers' conditions in Session 3 of the two treatments differ to each other both in terms of whether they have communicated with other workers about the work in previous tasks and whether they can communicate with other workers in the current tasks, making it difficult to draw any causal conclusions on the effect of peer communication.

[6] We count each chunk of sentences a worker entered in the chat room as a "turn."

each other before submitting their answers. Therefore, their answers might be correlated. Yet, existing aggregation methods all assume each worker complete the work independently, making it necessary for us to develop new ways to address the data correlation issue. Second, deploying peer communication incurs higher cost, since it requires us to hire workers in pairs to work for longer period of time and may need additional effort for worker synchronizations. Therefore, even though peer communication produces higher work quality for individual tasks, it is not clear deploying peer communication is always beneficial for the overall utility.

In this section, we focus on the setting in which a requester aims to collect labels for a batch of binary classification tasks with a fixed budget, and the "utility" to maximize is the average accuracy the requester obtains across all classification tasks[7]. In the following, we first discuss how to deal with the data correlation issue, that is, how to infer the correct label for a task given multiple labels solicited from workers, where some of the labels may be correlated. We then describe our algorithmic framework, a constrained Markov decision process, which adaptively decides whether and when peer communication should be deployed in each task under the budget constraint while taking into account data correlation and differing cost in deploying peer communication.

4.1 Dealing with Data Correlation

When peer communication is used in a task, a pair of workers directly interact with each other. Naturally, their contributions (e.g., labels in image labeling tasks) might be correlated. For categorical tasks with a finite number of labels, we could use the covariance notion to measure the correlation of workers' contributions. Formally, let X, Y be the random variables representing the answers generated by a pair of workers for the same task. The correlation of workers' answers can be formulated using covariance $cov(X, Y)$, defined as $cov(X, Y) = E[XY] - E[X]E[Y]$. By definition, when a pair of answers X, Y are independent, the covariance should be 0.

4.1.1 Measuring Data Correlation. To see whether the answers from a pair of workers are correlated when they work together on a task with peer communication, we examine workers' answers in Session 1 of both treatments in our experiments on image labeling tasks. For each of the 20 images in the experiments (with labels in $\{0, 1\}$), we calculate the covariance between pairs of labels generated in independent tasks (Session 1 of Treatment 1) and discussion tasks (Session 1 of Treatment 2)[8], in which we use the empirical average to replace the expectation in the definition. The results are shown in Figure 3. Perhaps not surprisingly, data collected in independent tasks is mostly independent (with covariance close to 0), while data collected in discussion tasks is correlated to various degrees.

We also calculate the covariance of workers' answers in Session 2 of both treatments to see if the data correlation caused by peer communication has any spillover effect. We find the covariance is close to 0 for both treatments, indicating the correlation in data caused by peer communication does not carry on to later independent work.

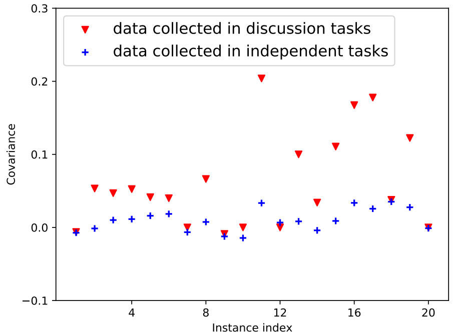


Figure 3: Covariance for data collected in independent tasks and discussion tasks in Session 1 in the image labeling HITs.

4.1.2 Meta-Workers and Meta-Labels. The above observations confirm that workers' answers are indeed correlated when peer communication is deployed. To deal with data correlation, we introduce the notions of *meta-workers* and *meta-labels*. In particular, we denote a pair of workers who talk with each other through the peer communication procedure as a meta-worker, and the pair of labels they generate as a meta-label. As no communication happens between different pairs of workers, we assume each meta-label is drawn independently.

Formally, for a binary classification task, let the true label $z \in \{0, 1\}$. When peer communication is deployed in a task, we obtain a pair of labels, which can be $\{1, 1\}$, $\{0, 1\}$, or $\{0, 0\}$, and we use the *meta-label* s_{11}, s_{01}, and s_{00} to denote them, respectively. Moreover, denote s_1 and s_0 as the label 1 and 0 obtained from a single worker who works independently.

To simplify the discussion, we assume workers are homogeneous. We propose a model to characterize the correlation in data produced in tasks with peer communication as follows: Denote p as the probability of independent workers providing correct labels, i.e., $p = P(s_1|z = 1) = P(s_0|z = 0)$. Additionally, we denote p_+, p_0, p_- as the probability for workers in tasks with peer communication to contribute two correct labels, one correct and one incorrect label, and two incorrect labels[9]:

$$p_+ = P(s_{11}|z = 1) = P(s_{00}|z = 0)$$
$$p_- = P(s_{00}|z = 1) = P(s_{11}|z = 0)$$
$$p_0 = P(s_{01}|z = 1) = P(s_{01}|z = 0)$$

This model provides a principled way to capture different levels of correlation. For example, when the pair of labels are independent, and the probability for each worker in the pair to submit a correct label is still p, we should have $p_+ = p^2, p_- = (1 - p)^2$, and $p_0 = 2p(1 - p)$. When the correlation between a pair of labels is 1 (i.e., the two labels are always the same), we have $p_0 = 0$.

4.1.3 Utilizing Meta-labels. The idea of introducing meta-workers and meta-labels are intuitive but powerful. Below we use maximum likelihood aggregation as an example to demonstrate how the

[7] Our discussion can be extended to classification tasks with any finite number of labels. Extending our results to general types of tasks (such as transcription tasks) requires a well-defined utility notion that can quantify the total utility for any given set of worker contributions. It is an interesting and important future direction.

[8] Recall that in our experiment, we always send a pair of workers to work on the same sequence of tasks. Thus, an independent task is also completed by a pair of workers except that they don't communicate with each other. This allows us to directly calculate the covariance for labels generated in independent tasks.

[9] This is the extension to the standard one-coin model in crowdsourcing literatures. Extending the discussion to model the confusion matrix (e.g., using two different probability values for $P(s_{11}|z = 1)$ and $P(s_{00}|z = 0)$) is straightforward. We do not include the discussion here due to space constraints.

concepts of meta-workers and meta-labels can be incorporated in standard aggregation methods, which provides us with key insights on how to utilize these concepts in our algorithmic framework (we will detail this in Section 4.2).

For a task with unknown true label $z \in \{0, 1\}$, given a set of N labels (or meta-labels) $\boldsymbol{L} = \{l_1, ..., l_N\}$, where $l_i \in \{s_{11}, s_1, s_{01}, s_0, s_{00}\}$, the maximum likelihood estimator for the value of z is defined as:

Definition 1. *Let the ground truth of the task be z. Given a set of labels* $\mathbf{L} = \{l_1, \ldots, l_N\}$. $\hat{z}$ *is a maximum likelihood estimator if*

$$\hat{z} = \begin{cases} 1 & \textit{if } P(\boldsymbol{L}|z=1) \geq P(\boldsymbol{L}|z=0), \\ 0 & \textit{otherwise.} \end{cases}$$

We assume p, p_+, p_0, and p_- are all known. Note that in our algorithmic framework as explained in Section 4.2, we adopt a Bayesian setting to learn how to aggregate the data over time without prior knowledge on values of these parameters. However, when such prior knowledge is available, a weighted majority voting rule can lead to maximum likelihood estimation:

Lemma 1. *Given a set of labels* $\mathbf{L}$. *Let* $n_{11}, n_1, n_{01}, n_0, n_{00}$ *denote the number of labels* $s_{11}, s_1, s_{01}, s_0, s_{00}$ *in* $\mathbf{L}$. *Consider the following weighted majority voting rule that generates an aggregation* $\hat{z}$

$$\hat{z} = \begin{cases} 1 & \textit{if } w_{11}n_{11} + w_1 n_1 \geq w_{00}n_{00} + w_0 n_0 \\ 0 & \textit{if } w_{11}n_{11} + w_1 n_1 < w_{00}n_{00} + w_0 n_0 \end{cases}$$

This weighted majority voting rule leads to maximum likelihood estimation when the weights are set as: $w_{11} = w_{00} = \ln \frac{p_+}{p_-}$, *and* $w_1 = w_0 = \ln \frac{p}{1-p}$.

Proof. We can write the probabilities on both sides as follows:

$$P(\boldsymbol{L}|z=1) = p_+^{n_{11}} p^{n_1} p_0^{n_{01}} (1-p)^{n_0} p_-^{n_{00}}$$

$$P(\boldsymbol{L}|z=0) = p_-^{n_{11}} (1-p)^{n_1} p_0^{n_{01}} p^{n_0} p_+^{n_{00}}$$

Therefore, we have

$$\frac{P(\boldsymbol{L}|z=1)}{P(\boldsymbol{L}|z=0)} = \left(\frac{p_+}{p_-}\right)^{n_{11}} \left(\frac{p}{1-p}\right)^{n_1} \left(\frac{1-p}{p}\right)^{n_0} \left(\frac{p_-}{p_+}\right)^{n_{00}}$$

Note that, in maximum likelihood estimator, $\hat{z} = 1$ if $P(\boldsymbol{L}|z = 1)/P(\boldsymbol{L}|z = 0) \geq 1$. Therefore, $\hat{z} = 1$ if

$$\left(\frac{p_+}{p_-}\right)^{n_{11}} \left(\frac{p}{1-p}\right)^{n_1} \geq \left(\frac{p}{1-p}\right)^{n_0} \left(\frac{p_+}{p_-}\right)^{n_{00}}$$

The proof is completed by taking logarithm on both sides. □

As a sanity check, we can see that when a pair of labels are independent (i.e., $p_+ = p^2$ and $p_- = (1-p)^2$), we have $w_{11} = w_{00} = 2w_1 = 2w_0$, implying that the weight of $\{1, 1\}$ label is twice the weight of $\{1\}$ label, and this is essentially a simple majority voting.

Note that in the maximum likelihood aggregation, the number of meta-label s_{01} does *not* play a role in the aggregation process. In other words, we may interpret the generation of meta-labels as follows: with probability p_+ (or p_-), a meta-worker generates a correct label s_{11} (or incorrect label s_{00}), while with probability p_0 she generates *no* label at all. The above weighted majority voting rule then simply indicates that different weights need to be used for labels generated by independent workers or meta-workers. Following a similar idea, in the following algorithmic framework, we only take the meta-label s_{00} and s_{11} into consideration and discard the meta-label s_{01} when inferring the correct labels of a task from a collection of labels and meta-labels.

4.2 Our Algorithmic Framework

With the notions of meta-workers and meta-labels in place, we have a principled way to deal with correlated data in peer communication. However, we still need to address the second challenge of balancing the quality and cost. In particular, while introducing peer communication leads to a significant improvement in work quality for individual microtasks, such improvement comes with extra cost, such as the financial payment incurred to recruit more workers (e.g., at least two workers are needed for peer communication to happen), the compensation for longer task completion time due to discussions, and the additional administrative costs for synchronizing the work pace of worker pairs. As a result, a requester needs to face the quality-cost tradeoff when deploying peer communication.

We now describe our algorithmic framework, built on the constrained Markov decision process (CMDP), that adaptively decides for a requester with a limited budget, whether and when peer communication should be deployed in each of his tasks with the goal of maximizing his total utility (i.e., the average accuracy for all classification tasks), while taking into account data correlation and differing costs for deploying peer communication.

4.2.1 Problem Setup. Our problem setup is inspired by the method by Chen et al. [8] to optimally allocate budget among task instances in crowdsourcing data collection. Our setup differs from theirs in two fundamental ways due to the presence of peer communication strategy. First, they don't and don't need to consider the issue of data correlation. Second, in their setting, the cost for acquiring labels is fixed, while we need to deal with the differing costs when peer communication is deployed in a task. Therefore, instead of modeling the decision-making problem as a Markov decision process framework (as in Chen et al. [8]), we adopt a constrained Markov decision process framework and include the meta-label concept in our formulation.

Formally, suppose a requester gets a budget of $\mathcal{B}$ and a batch of K binary classification tasks, and he needs to estimate the label for each of these tasks. The goal of the requester is to maximize the average accuracy of the estimated labels across all tasks through spending the budget to solicit labels from crowd workers and then aggregating the collected labels. We describe the setting in which workers are homogeneous (however, their performance might be different when working independently or when working with peer communication). Extensions to settings with heterogeneous workers are straightforward as described by Chen et al. [8].

Assume the K tasks are independent from each other, and $Z_k \in \{0, 1\}$ represents the true label for task k ($1 \leq k \leq K$). We use the notations $\theta_k \in [0, 1]$, $\alpha_s \in [0, 1]$, and $\alpha_p \in [0, 1]$ to model the label generation process, where θ_k characterizes the difficulty of task k, α_s and α_p characterize workers' performance when working independently and working with peer communication. In particular, we denote $p_{k,s,1}$ (or $p_{k,s,0}$) as the probability for a single worker (who works independently) to provide label 1 (or 0) for task k. We define $p_{k,s,1} = \alpha_s \theta_k + (1 - \alpha_s)(1 - \theta_k)$ and $p_{k,s,0} = 1 - p_{k,s,1}$. To obtain intuition for the parameters of the model, assume $\alpha_s = 1$, we can see that θ_k captures the difficulty of task k: When θ_k is close to 0.5, workers are effectively making random guess (hence the task is difficult), and when θ_k is close to 0 or 1, independent workers can consistently provide the same label (hence the task is easy). Similarly, α_s can then be interpreted as the worker skill and a larger α_s implies a higher skill. We assume θ_k is consistent with the label Z_k, which means $Z_k = 1$ if and only if $\theta_k \geq 0.5$.

Recall that we denote a meta-worker as a pair of workers in tasks with peer communication. We use α_p to denote the skill of meta-workers, and the probability for a meta-worker to generate a meta-label s_{01} for task k is denoted as q_k. Conditioned on a meta-worker contributing a meta-label other than s_{01}, α_p is similarly defined as α_s. That is, when $p_{k,p,1}$ and $p_{k,p,0}$ is the probability for a meta-worker to generate meta-labels s_{11} and s_{00}, we have $p_{k,p,1} = (1-q_k)(\alpha_p\theta_k + (1-\alpha_p)(1-\theta_k))$ and $p_{k,p,0} = 1-p_{k,p,1}-q_k$.

After describing the data generation model, we formulate the online decision problem faced by the requester. The requester recruits workers to label his tasks in a sequential manner. Specifically, at each time step t, the requester decides on a task k_t to work on, and he can solicit label(s) from crowd workers on this task using one of the two strategies (the strategy is denoted as x_t): first, the requester can recruit a *single* worker to work on the task ($x_t = 0$), and thus obtain a label for that task; second, the requester may recruit a meta-worker (i.e., a *pair* of workers following the peer communication procedure) to work on the task ($x_t = 1$), and thus obtain a meta-label for the task. We denote c_s as the cost for recruiting a single worker and c_p ($c_p > c_s$) as the cost of recruiting a meta-worker through peer communication strategy.

Naturally, the requester's activity in each time step can be summarized through the tuple (k_t, x_t). We also denote y_t as the label (or meta-label) obtained by the requester at time t for task k_t. By the time $t_\mathcal{B}$ that the requester exhausts his budget, his activity history is $\mathcal{H}_\mathcal{B} = \{(k_0, x_0, y_0), ..., (k_{t_\mathcal{B}}, x_{t_\mathcal{B}}, y_{t_\mathcal{B}})\}$. The requester then aggregates the data he has collected and infers the true labels for each of the K tasks such that the expected accuracy across all K tasks, conditioned on the activity history $\mathcal{H}_\mathcal{B}$, is maximized.

4.2.2 A Constrained Markov Decision Process Formulation. We now formally model the requester's decision-making problem as a constrained Markov decision process:

- **States**: the state s_t is a $K \times 4$ matrix, where $s_t(k, \cdot)$ is a 1×4 vector with each entry representing before time t, the number of label (or meta-labels) s_0, s_1, s_{00}, s_{11} obtained for task k. Note that following the idea that we have discussed in Section 4.1.3, we consider the meta-label s_{01} to contributes zero utility to the requester and thus we do not include the count of it in the state.
- **Actions**: $a_t = (k_t, x_t)$, where k_t is the task to work on at time t, and $x_t \in \{0, 1\}$ represents the worker recruiting strategy, with 0 being recruiting a single worker working independently and 1 being recruiting a pair of workers to follow the peer communication procedure.
- **Transition probabilities**: When $a_t = (k_t, x_t = 0)$,

$$Pr(s_{t+1}|s_t, a_t) = \begin{cases} p_{k_t,s,1} & \text{if } s_{t+1} = s_t + (\mathbf{0}, \mathbf{e_{k_t}}, \mathbf{0}, \mathbf{0}) \\ p_{k_t,s,0} & \text{if } s_{t+1} = s_t + (\mathbf{e_{k_t}}, \mathbf{0}, \mathbf{0}, \mathbf{0}) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{e_{k_t}}$ is a $K \times 1$ vector with value 1 at the k_t-th entry and 0 at all other entries. On the other hand, when $a_t = (k_t, x_t = 1)$,

$$Pr(s_{t+1}|s_t, a_t) = \begin{cases} p_{k_t,p,1} & \text{if } s_{t+1} = s_t + (\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{e_{k_t}}) \\ 1 - p_{k_t,p,1} - q_{k_t} & \text{if } s_{t+1} = s_t + (\mathbf{0}, \mathbf{0}, \mathbf{e_{k_t}}, \mathbf{0}) \\ q_{k_t} & s_{t+1} = s_t \\ 0 & \text{otherwise} \end{cases}$$

- **Rewards**: We adopt the same reward function as that used by Chen et al. [8]. Specifically, we assume the parameters $\theta_k, \alpha_s, \alpha_p$ are sampled from three separate Beta prior distributions, and we update the posteriors of these distributions through variational approximation where hyper-parameters are decided by moment matching. Doing so, we can then define the reward as $R(s_t, a_t) = \mathbb{E}(h(P_{k_t}^{t+1}) - h(P_{k_t}^{t}))$, where P_k^t is the probability of the parameter θ_k taking on a value of at least 0.5 given the posterior of θ_k at time t, $h(x) = \max(x, 1-x)$, and the expectation is taken over all possible label y_t observed after action a_t.
- **Constraint**: Different from the setting in the work by Chen et al. [8], as different actions imply different costs, we need to explicitly characterize the budget constraint for our problem. Formally, the requester needs to ensure the budget constraint is satisfied. $\sum_{t=0}^{t_\mathcal{B}} c_s \mathbf{1}(x_t = 0) + c_p \mathbf{1}(x_t = 1) \leq \mathcal{B}$, where $\mathbf{1}(.)$ is the indicator function.

4.2.3 Proposed Algorithm. We adopt the method of Lagrangian multipliers to solve the above constrained optimization problem, which converts the problem of maximizing the total reward (i.e., $\sum_{t=0}^{t_\mathcal{B}} R(s_t, a_t)$) under the budget constraint into a simpler problem of maximizing the auxiliary function $\sum_{t=0}^{t_\mathcal{B}} R(s_t, a_t) - \lambda \sum_{t=0}^{t_\mathcal{B}} (c_s \mathbf{1}(x_t = 0) + c_p \mathbf{1}(x_t = 1))$. Notice this optimization problem is equivalent to solve a (unconstrained) Markov decision process where reward in each step is redefined as $R'(s_t, a_t) = R(s_t, a_t) - \lambda(c_s \mathbf{1}(x_t = 0) + c_p \mathbf{1}(x_t = 1))$. We use the optimistic knowledge gradient technique introduced by Chen et al. [8] to solve the optimal policy of this MDP, which produces a single-step look-ahead policy that maximizes the highest reward at each step. Note that in theory, we can compute the optimal value of λ by solving the dual of the constrained MDP. In practice, we have experimented with multiple different λ values and find that the choice of λ has limited influence on the performance of our algorithmic approach.

4.3 Evaluations

We evaluate the effectiveness of our algorithmic approach using both real-world data and synthetic data.

4.3.1 Experiments on Real Data. Using the real data that we collected in image labeling tasks of our experimental study, we compare the performance of our algorithm with a couple of baseline algorithms. In our evaluation, we set the cost of recruiting a single worker as $c_s = 1.0$ and the cost of recruiting a pair of workers to work on a task with peer communication (i.e., a meta-worker) $c_p = 2.5$. Note that we have examined a range of different values of c_p from 1.5 to 3.5 and the results are qualitatively similar. The prior distribution for θ_k is set as Beta(1, 1), where the prior distributions for α_s and α_p are all set to be Beta(4, 1). For this evaluation, we only considered the final labels that workers in our experimental study submit in the *first two tasks* of the image labeling HIT[10]. Thus, when $a_t = (k_t, 0)$, we randomly sampled a label from Treatment 1 workers who had completed task k_t in their first two (independent) tasks, and when $a_t = (k_t, 1)$, we randomly sampled a label from Treatment 2 workers who had completed task k_t in their first two (discussion) tasks.

The performance of our algorithmic approach is compared against the following baseline approaches:

- *Round robin*: in each round, the requester decides which task to work on in a round robin fashion, and he always recruit a single worker to work on that task independently.
- *Single workers only*: in each round, the requester recruits a single worker to work on a task independently, and this task is optimally decided (effectively by considering only actions with $x_t = 0$ in our algorithm).
- *Peer communication only*: in each round, the requester recruits a pair of workers to work on a task with peer communication, and this task is optimally decided (effectively by considering only actions with $x_t = 1$ in our algorithm).
- *Our algorithm [No correlation]*: in each round, the requester uses our algorithm to decide whether to deploy peer communication and which task to work. The only difference is that this baseline treats the two labels from peer communication as two independent labels while our algorithm incorporates the concept of meta-labels to deal with data correlation.

We conduct this evaluation on a range of budget level from 20 to 400 with an interval of 20. At each budget level, we implement each of the decision-making strategies for 100 times, and we report the average level of overall accuracy the requester obtains across the 20 tasks when she exhausts the budget in Figure 4.

As shown in Figure 4, our proposed algorithm outperforms all baseline strategies. In particular, we make a few observations as follows. First, comparing the performance of our algorithm and that of the "No Correlation" strategy, it is clear that incorporating meta-labels to deal with data correlation has improved the requester's overall utility. In fact, even for the "peer communication only" strategy, we also implement two versions, and the version for which the idea of meta-labels is used also outperforms the other version treating two labels generated by pairs of workers as independent. In the following discussion, unless otherwise specified, we adopt the meta-worker ideas in our implementation when peer communication is used. Second, strategies involving peer communication converge to a better overall accuracy than strategies without peer communication does. This is due to the fact that for some tasks in our experiment, the majority of workers who work independently provide incorrect answers while the majority of workers with peer communication provide correct answers. Third, adaptively determining whether and when to deploy peer communication outperforms fixed recruiting strategies, as illustrated by the superior performance of our algorithm over both the "single workers only" and "peer communication only" strategies. Finally, adaptively deciding which task to label next significantly improves the total utility than random task assignment does, e.g., through observing the significantly worse performance of the baseline "round robin" strategy.

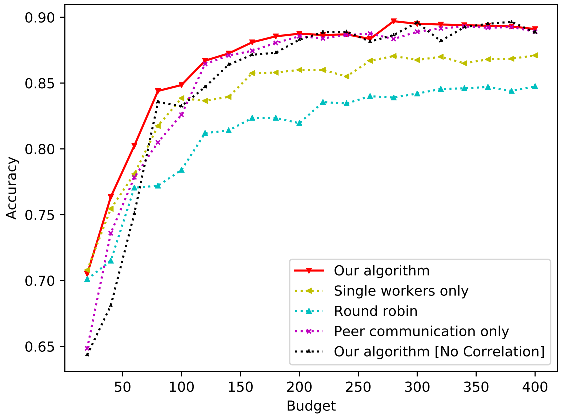


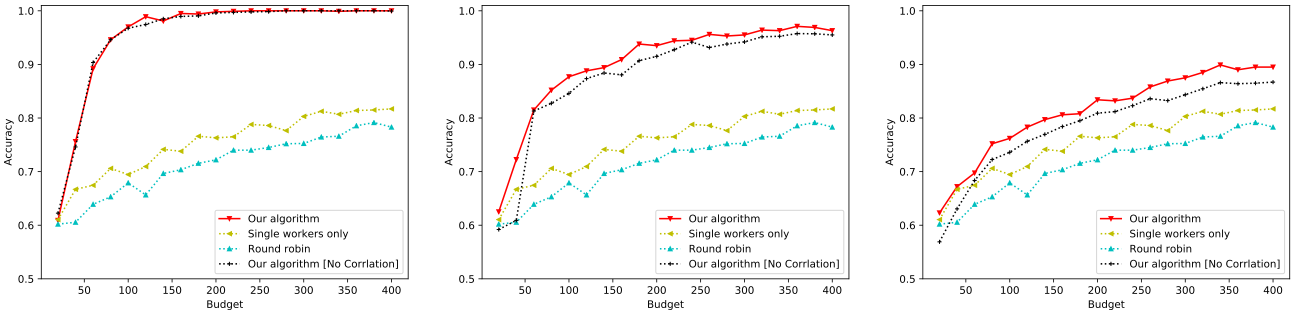
Figure 4: Evaluating the performance of the proposed approach on real datasets.

4.3.2 Experiments on Synthetic Data. To the best of our knowledge, our dataset is the only dataset that deploys peer communication for crowdsourcing data collection. Therefore, to further investigate the properties of our proposed algorithm, we generate synthetic data to evaluate our algorithm. In particular, we explore how the performance of our algorithm changes along the following three dimensions: 1) the level of data correlation of workers' answers in tasks with peer communication, 2) the performance gap between workers who work independently and workers who discuss with others via peer communication, and 3) the cost differences of hiring a single worker and hiring a pair of workers for peer communication. In the base setup, we set θ_k to be uniformly drawn from $[0.5, 1]$, α_s drawn from a normal distribution with mean 0.7 and variance 0.01, α_p drawn from a normal distribution with mean 0.9 and variance 0.1. We also set $c_s = 1$ and $c_p = 2.5$.

We first modify the level of data correlation of workers' answers in peer communication. This can be done by changing the value of q_k, i.e., the probability of a pair of workers in peer communication to generate the meta-label s_{01}. In strong correlation, we set $q_k = 0$, which means the two workers are entirely correlated (always generating the same label). In no correlation, we set q_k to the value such that the two labels in a meta-label are independently generated (i.e., $q_k = 2\sqrt{p_{k,p,1}p_{k,p,0}}$, which can be derived using our model discussed in Section 4.1.2). In weak correlation, q_k is uniformly drawn between the above two values. We compare the performance between our algorithm and "our algorithm [no correlation]", which treats the two labels from a meta-label as independent labels. As shown in Figure 5, the performance gap becomes larger as the correlation becomes stronger[11]. This validates the benefits of incorporating meta-labels in our framework when there is data correlation.

[10] Recall that we set out to examine the effects of peer communication using the first two tasks in each HIT.

[11] As a side note, the overall performance is lower in strong correlation since we fixed α_p in all three plots; two independent labels brings more information than two correlated labels. Since our goal is to measure the gap between two algorithms, we didn't tune the parameter to normalize the algorithm performance.



(a) No Correlation (b) Weak Correlation (c) Strong Correlation

Figure 5: The performance comparison under different levels of correlation in peer communication.

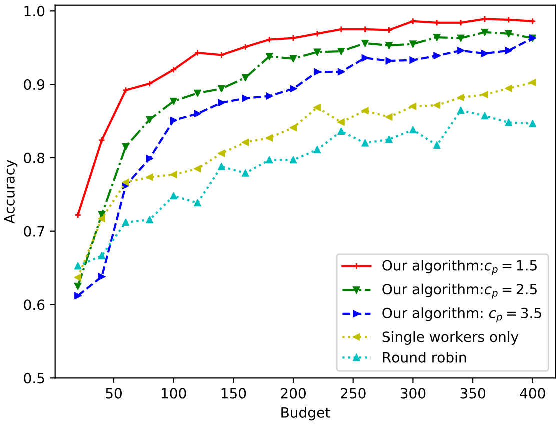


Figure 6: Modify the cost of peer communication.

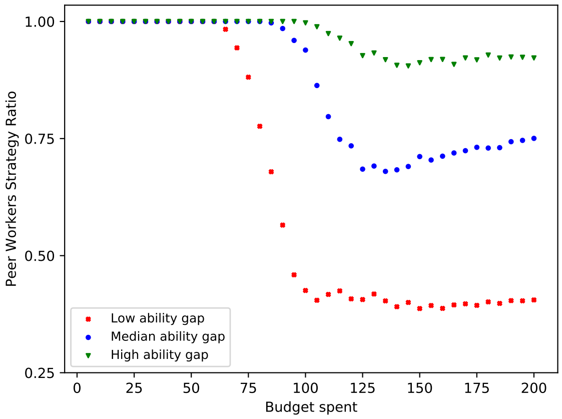


Figure 7: Ratio of peer communication strategies deployed.

We then change the cost of deploying peer communication c_p to be from $\{1.5, 2.5, 3.5\}$. As shown in Figure 6, our algorithm performance decreases as c_p increases. However, even when the cost of peer communication is pretty large (i.e., $c_p = 3.5$), utilizing peer communication is still beneficial. Next, we vary the performance gap between single workers (who work independently) and meta-workers by fixing the mean of α_p to be 0.9 and set the mean of α_s to be 0.8, 0.7, and 0.6. The results are qualitatively similar to changing c_p (e.g., larger skill gap corresponds to smaller c_p). To provide more insights for our algorithm, we demonstrate this result in a different plot. In particular, we fix the budget to be 200 and run our algorithm 100 times. We record the worker recruiting strategy (hiring single workers or deploying peer communication) our algorithm takes at every step, and then calculate the ratio of peer communication strategy as a function of the budget spent so far. As shown in Figure 7, our algorithm always starts by deploying peer communication. When the marginal rewards for hiring peer communication is not high enough to justify the higher cost, our algorithm gradually switches to hire single workers. These two figures demonstrate that our algorithm brings in benefits under a wide range of settings and has stronger benefits when c_p is small or when the performance gap between single workers and meta-workers (with peer communication) are higher.

5 CONCLUSION

In this paper, we relax the common assumption of data independence in crowdsouring data collection. In particular we explore peer communication, in which a pair of crowd workers directly communicate when producing the data. We first examine the effects of peer communication on the quality of crowd work produced. Randomized experiments conducted on Amazon Mechanical Turk demonstrate that the work quality significantly improves in tasks with peer communication. We then study how to utilize peer communication in crowdsourcing data collection. In particular, we introduce the notions of meta-workers and meta-labels to deal with data correlation caused by peer communication. We then develop an algorithmic framework, built on constrained Markov decision process, to optimally determine whether and when to deploy peer communication in crowdsourcing data collection, with the goal of maximizing the total utility of requesters while satisfying budget constraints. Experiments conducted on both real data and synthetic data demonstrate the advantage of our proposed algorithms over baseline approaches.

Our results suggest the potential benefits of incorporating peer communication in crowdsourcing and provide a framework for better utilizing these benefits. We hope this work could open more discussions on designing and leveraging more complex, useful worker interactions to further enhance crowdsourcing.

REFERENCES

[1] Alan Aipe and Ujwal Gadiraju. 2018. SimilarHITs: Revealing the Role of Task Similarity in Microtask Crowdsourcing. In *Proceedings of the 29th on Hypertext and Social Media*.

[2] Bahador Bahrami, Karsten Olsen, Peter E Latham, Andreas Roepstorff, Geraint Rees, and Chris D Frith. 2010. Optimally interacting minds. *Science* 329, 5995 (2010), 1081–1085.

[3] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 2011. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[4] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 313–322.

[5] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. 2010. VizWiz: Nearly Real-time Answers to Visual Questions. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[6] Carrie J. Cai, Shamsi T. Iqbal, and Jaime Teevan. 2016. Chain Reactions: The Impact of Order on Microtask Chains. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*.

[7] Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI)*.

[8] Xi Chen, Qihang Lin, and Dengyong Zhou. 2013. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *International Conference on Machine Learning (ICML)*.

[9] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1999–2008.

[10] Sharath R. Cholleti, Sally A. Goldman, Avrim Blum, David G. Politte, and Steven Don. 2008. Veritas: Combining expert opinions without labeled data. In *Proceedings 20th IEEE international Conference on Tools with Artificial intelligence (ICTAI)*.

[11] Derrick Coetzee, Seongtaek Lim, Armando Fox, Bjorn Hartmann, and Marti A Hearst. 2015. Structuring interactions for large-scale synchronous peer learning. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 1139–1152.

[12] Catherine Crouch and Eric Mazur. 2001. Peer instruction: Ten years of experience and results. *Am. J. Phys.* 69, 9 (September 2001), 970–977.

[13] Peng Dai, Christopher H. Lin, Mausam, and Daniel S. Weld. 2013. POMDP-based Control of Workflows for Crowdsourcing. *Artif. Intell.* 202, 1 (Sept. 2013), 52–85.

[14] A. P. Dawid and A. M. Skene. 1979. Maximum likeihood estimation of observer error-rates using the EM algorithm. *Applied Statistics* 28 (1979), 20–28.

[15] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B* 39 (1977), 1–38.

[16] Louis Deslauriers, Ellen Schelew, and Carl Wieman. 2011. Improved learning in a large-enrollment physics class. *science* 332, 6031 (2011), 862–864.

[17] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G Ipeirotis, and Philippe Cudré-Mauroux. 2015. The dynamics of micro-task crowdsourcing: The case of amazon mturk. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 238–247.

[18] Ryan Drapeau, Lydia B. Chilton, Jonathan Bragg, and Daniel S. Weld. 2016. MicroTalk: Using Argumentation to Improve Crowdsourcing Accuracy. In *Fourth AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.

[19] Mary L Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. 2016. The crowd is a collaborative network. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. ACM, 134–147.

[20] Neha Gupta, David Martin, Benjamin V Hanrahan, and Jacki O'Neill. 2014. Turk-life in India. In *Proceedings of the 18th International Conference on Supporting Group Work*. ACM, 1–11.

[21] Danna Gurari and Kristen Grauman. 2017. CrowdVerge: Predicting If People Will Agree on the Answer to a Visual Question. In *Proceedings of the 2017 Conference on Human Factors in Computing Systems (CHI)*.

[22] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. 2013. Adaptive Task Assignment for Crowdsourced Classification. In *The 30th International Conference on Machine Learning (ICML)*.

[23] Chien-Ju Ho, Aleksandrs Slivkins, Siddharth Suri, and Jennifer Wortman Vaughan. 2015. Incentivizing High Quality Crowdwork. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*.

[24] John Joseph Horton and Lydia B. Chilton. 2010. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce (EC)*.

[25] Lilly C Irani and M Silberman. 2013. Turkopticon: Interrupting worker invisibility in amazon mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 611–620.

[26] Rong Jin and Zoubin Ghahramani. 2003. Learning with multiple labels. In *Advances in Neural Information Processing Systems (NIPS)*.

[27] David R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Budget-Optimal Crowdsourcing using Low-rank Matrix Approximations. In *Proc. 49th Annual Conference on Communication, Control, and Computing (Allerton)*.

[28] David. R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *The 25th Annual Conference on Neural Information Processing Systems (NIPS)*.

[29] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. 2011. Novel Dataset for Fine-Grained Image Categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO.

[30] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J Guo, Robert C Miller, and Krzysztof Z Gajos. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 4017–4026.

[31] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[32] Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively Crowdsourcing Workflows with Turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work CSCW*.

[33] Edith Law, Ming Yin, Joslin Goh, Kevin Chen, Michael A. Terry, and Krzysztof Z. Gajos. 2016. Curiosity Killed the Cat, but Makes Crowdwork Better. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI)*.

[34] Qi Li, Fenglong Ma, Jing Gao, Lu Su, and Christopher J Quinn. [n. d.]. Crowdsourcing high quality labels with a tight budget. In *Proceedings of the ninth acm international conference on Web Search and Data Mining (WSDM)*.

[35] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[36] Frank Lyman. 1987. Think-Pair-Share: An Expanding Teaching Technique. *MAA-CIE Cooperative News* (1987).

[37] David Martin, Benjamin V Hanrahan, Jacki O'Neill, and Neha Gupta. 2014. Being a turker. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 224–235.

[38] Winter Mason and Duncane Watts. 2009. Financial Incentives and the "Performance of Crowds". In *Proceedings of the 1st Human Computation Workshop (HCOMP)*.

[39] Edward Newell and Derek Ruths. 2016. How One Microtask Affects Another. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*.

[40] Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 1–12.

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

[42] Daniela Retelny, Sébastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. 2014. Expert Crowdsourcing with Flash Teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[43] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. *ICWSM* 11 (2011), 17–21.

[44] Niloufar Salehi, Lilly C Irani, Michael S Bernstein, Ali Alkhatib, Eva Ogbe, Kristy Milland, et al. 2015. We are dynamo: Overcoming stalling and friction in collective action for crowd workers. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, 1621–1630.

[45] Mike Schaekermann, Joslin Goh, Kate Larson, and Edith Law. 2018. Resolvable vs. Irresolvable Disagreement: A Study on Worker Deliberation in Crowd Work. In *Proceedings of the 21st ACM Conference on Computer Supported Cooperative Work &#38; Social Computing (CSCW)*.

[46] Nihar Bhadresh Shah and Denny Zhou. 2015. Double or Nothing: Multiplicative Incentive Mechanisms for Crowdsourcing. In *The 29th Annual Conference on Neural Information Processing Systems (NIPS)*.

[47] Aaron D. Shaw, John J. Horton, and Daniel L. Chen. 2011. Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work (CSCW)*.

[48] Michelle K Smith, William B Wood, Wendy K Adams, Carl Wieman, Jennifer K Knight, Nancy Guild, and Tin Tin Su. 2009. Why peer discussion improves student performance on in-class concept questions. *Science* 323, 5910 (2009), 122–124.

[49] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels

from labelers of unknown expertise. In *Advances in Neural Information Processing Systems (NIPS)*.

[50] Ming Yin, Yiling Chen, and Yu-An Sun. 2013. The Effects of Performance-Contingent Financial Incentives in Online Labor Markets. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*.

[51] Ming Yin, Mary L Gray, Siddharth Suri, and Jennifer Wortman Vaughan. 2016. The communication network within the crowd. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1293–1303.

[52] Lixiu Yu, Paul André, Aniket Kittur, and Robert Kraut. 2014. A comparison of social, learning, and financial strategies on crowd engagement and output quality. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 967–978.

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