iCrowd: An Adaptive Crowdsourcing Framework

Ju Fan[‡] Guoliang Li[†] Beng Chin Ooi[‡] Kian-lee Tan[‡] Jianhua Feng[†]

[‡]School of Computing, National University of Singapore, Singapore.
[†]Department of Computer Science, TNList, Tsinghua University, Beijing, China.
{fanj, ooibc, tankl}@comp.nus.edu.sg; {liguoliang,fengjh}@tsinghua.edu.cn

ABSTRACT

Crowdsourcing is widely accepted as a means for resolving tasks that machines are not good at. Unfortunately, Crowdsourcing may yield relatively low-quality results if there is no proper quality control. Although previous studies attempt to eliminate "bad" workers by using qualification tests, the accuracies estimated from qualifications may not be accurate, because workers have diverse accuracies across tasks. Thus, the quality of the results could be further improved by selectively assigning tasks to the workers who are well acquainted with the tasks. To this end, we propose an adaptive crowdsourcing framework, called *i*`Crowd`. *i*`Crowd` on-the-fly estimates accuracies of a worker by evaluating her performance on the completed tasks, and predicts which tasks the worker is well acquainted with. When a worker requests for a task, *i*`Crowd` assigns her a task, to which the worker has the highest estimated accuracy among all online workers. Once a worker submits an answer to a task, *i*`Crowd` analyzes her answer and adjusts estimation of her accuracies to improve subsequent task assignments. This paper studies the challenges that arise in *i*`Crowd`. The first is how to estimate diverse accuracies of a worker based on her completed tasks. The second is instant task assignment. We deploy *i*`Crowd` on Amazon Mechanical Turk, and conduct extensive experiments on real datasets. Experimental results show that *i*`Crowd` achieves higher quality than existing approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

Keywords

Crowdsourcing; Quality control; Adaptive task assignment

1. INTRODUCTION

Crowdsourcing outsources tasks for solutions from an unknown group of people (aka workers), which is indeed useful to many real-world applications, e.g., image search, entity resolution, answering database-hard queries [24, 23, 27, 28, 12, 32]. Due to its openness, crowdsourcing yields relatively low-quality results, or even noise, which attracts great interest in devising good quality control methods [17]. Existing methods [32, 18, 30] employ a *redundancy*-based strategy which publishes a crowdsourcing task to multiple workers and derives the result by aggregating worker answers. A naïve aggregation approach is majority voting that chooses the answer that the majority of the workers yield as the result. Recently, more sophisticated approaches have been proposed and they can be broadly classified into two categories. The *gold-injected* approaches [22] leverage small amounts of tasks with ground truth to estimate workers' quality, while the *EM*-based approaches [31, 8] simultaneously estimate worker quality and predict aggregated results using an Expectation-Maximization (EM) strategy. Moreover, existing approaches further improve the quality by eliminating "bad" workers. A well-known way is to use *qualification tests* to distinguish bad and good workers, and stop assigning tasks to workers that cannot give good answers to the qualification tasks.

Although existing methods perform well in simple crowdsourcing tasks, such as image labeling, they may have limitations on more complicated crowdsourcing tasks that require domain knowledge. In these tasks, workers may have diverse accuracies across tasks, as they are usually good at tasks in domains they are familiar with but may provide low-quality answers in unfamiliar domains. Take crowdsourced entity resolution [32] as an example. A worker acquainted with Samsung stands a better chance to correctly differentiate the models "Note4" and "S4", while she may not be good at tasks about iPad and cannot identify that "iPad with Retina display" is colloquially referred to as "iPad 4". Workers with different backgrounds may be good at different topics: a basketball fan stands a better chance to correctly annotate tables related to NBA, while a film enthusiast is more reliable for tables involving Hollywood films. Similar observations can be found in other tasks. We have conducted empirical investigation on two complicated crowdsourcing tasks, evaluating quality of Yahoo Answers and comparing items (e.g., which car is more fuel efficient), and report empirical observations of accuracy diversity in Figure 6.

The accuracy diversity in crowdsourcing engenders many challenges, making existing solutions inadequate for producing high-quality result. On the one hand, a worker, who gives good answers to qualification tests, may not provide promising answers to other assigned tasks. As such, the existing approaches may over- or under-estimate workers' ac-

SIGMOD'15, May 31 – Jun 4, 2015, Melbourne, Victoria, Australia.

http://dx.doi.org/10.1145/2723372.2750550.

Table 1: Microtasks for verifying whether two entities are matched.

Microtask	Verifying two entities	Tokens
t_1	(iphone 4 WiFi 32GB, iphone four 3G black)	{iphone 4 WiFi 32GB four 3G black}
t_2	(ipod touch 32GB WiFi, ipod touch headphone)	{ipod touch 32GB WiFi headphone}
t_3	(ipad 3 WiFi 32GB black, new ipad cover white)	{ipad 3 WiFi 32GB black new cover white}
t_4	(iphone four WiFi 16GB, iphone four 3G 16GB)	{iphone four WiFi 16GB 3G}
t_5	(iphone 4 case black, iphone 4 WiFi 32GB)	{iphone 4 case black WiFi 32GB}
t_6	(iphone 4 WiFi 32GB, iphone four WiFi 32GB)	{iphone 4 WiFi 32GB four}
t_7	(ipod touch 32GB WiFi, ipod touch case black)	{ipod touch 32GB WiFi case black}
t_8	(ipod touch headphone, ipod nano headphone)	{ipod touch nano headphone}
t_9	(ipod touch WiFi, ipod nano headphone)	{ipod touch WiFi nano headphone}
t_{10}	(ipad 3 WiFi 32GB black, iphone 4 cover white)	{ipad 3 WiFi 32GB black iphone 4 cover white}
t_{11}	(ipad 4 WiFi 16GB, ipad retina display WiFi 16GB)	{ipad 4 WiFi 16GB retina display}
t_{12}	(ipad 3 cover white, new ipad cover white)	{ipad 3 cover white new}

curacies and thus result in unreliable aggregated results. On the other hand, existing approaches neglect a fact that we can *adaptively* assign tasks to workers who have expertise on the tasks to further improve the quality, instead of random task assignment without considering workers' expertise.

To address the limitations of existing approaches, we propose an adaptive crowdsourcing framework, called *i*`Crowd`. *i*`Crowd` on-the-fly estimates accuracies of a worker by evaluating her performance on the completed tasks, and infers worker's accuracies on similar tasks. When a worker requests for a task, the framework assigns the worker a task, to which the worker has the highest estimated accuracy among all online workers. Once a worker submits her answer to a task, *i*`Crowd` analyzes her answer and adaptively adjusts the accuracy estimation to improve any subsequent task assignments. In this way, *i*`Crowd` can effectively predict which workers are more appropriate for a task, and adaptively assigns the task to these high-quality workers.

We address two main research challenges that arise in adaptive crowdsourcing. The first one is how to estimate the *diverse* accuracies of workers based on their completed tasks. To address this challenge, we propose an accuracy estimation method by considering the "similarity" of tasks: a worker may have comparable accuracies on tasks in similar domains. We first construct a graph to model similarity of tasks and evaluate worker accuracies on her completed tasks. The second one is instant task assignment based on the estimated accuracies. As workers are generally impatient to wait for too long for a task assignment, we need to efficiently assign tasks to the workers. We develop efficient algorithms to support instant task assignment. Since existing platforms, such as Amazon Mechanical Turk (AMT) [2], have no functionality to support assigning tasks to workers, we develop an *i*`Crowd` system which iteratively communicates with the platforms to receive task requests from workers, assign tasks to them, and obtain answers from the workers.

To summarize, we make the following contributions.

(1) We formulate the problem of adaptive crowdsourcing and develop a framework *i*`Crowd` to support adaptive crowdsourcing in existing crowdsourcing platforms (see Section 2).

(2) We propose a graph-based estimation model to estimate the accuracies of a worker based on her completed tasks, which can tackle the diverse accuracies of workers across tasks and provide accurate estimation (see Section 3).

(3) We devise an adaptive assignment framework, prove that the optimal task assignment problem is NP-hard, and develop a greedy algorithm to enable instant task assignments (see Section 4).

(4) We deploy *i*`Crowd` on AMT and conduct extensive experiments on two real datasets. Experimental results show that *i*`Crowd` achieves 10% - 20% improvement on accuracy compared with state-of-the-art approaches (see Section 6).

2. AN OVERVIEW OF ADAPTIVE CROWDSOURCING

2.1 Problem Statement

Microtasks. Consider a requester who publishes a set of microtasks $\mathcal{T} = \{t_1, t_2, \ldots t_m\}$. For ease of presentation, each microtask is a binary microtask with `YES/NO` choices. Note that our techniques can be extended to microtasks with more than two choices. Table 1 provides twelve microtasks for *entity resolution*. Each microtask wants workers to verify whether two records (in the second column) are matched as a same product model. For example, t_1 requires workers to verify whether "`iphone 4 WiFi 32GB`" and "`iphone four 3G black`" are duplicated models. The worker, who has been assigned with t_1, would answer `YES` if she agrees that they are the same product, or `NO` otherwise.

Workers. A set of workers $\mathcal{W} = \{w_1, w_2, \ldots, w_n\}$ will work on microtasks in $\mathcal{T}$. Note that worker set in crowdsourcing is *dynamic*: any existing worker may become *inactive* by stopping work on $\mathcal{T}$ while new workers may become *active*. Moreover, since workers are prone to errors [22], answers provided by them may not be always correct. To predict whether a worker can correctly answer a microtask, we introduce *accuracy* defined as below.

Definition 1 (Accuracy). *The accuracy of a worker $w \in \mathcal{W}$ on a microtask $t_i \in \mathcal{T}$, denoted by p_i^w, is the probability $p_i^w = Pr\{w$ correctly answers $t_i\}$.*

For simplicity, we use vector $\mathbf{p}^w = \{p_1^w, p_2^w, \ldots, p_{|\mathcal{T}|}^w\}$ to represent the accuracies of w on microtasks in $\mathcal{T}$.

Microtask Assignment. In crowdsourcing, to improve the quality, a microtask is usually assigned to multiple workers and its result is obtained via a voting scheme. Under this scheme, we assign a microtask t_i to a worker set $\mathcal{W}_{t_i} \subset \mathcal{W}$ with size k, where k is an *assignment size* to represent the number of workers that can be assigned with t_i. k is usually provided by the crowdsourcing requester. Given worker set $\mathcal{W}_{t_i}$, we utilize (weighted) majority voting, which is well accepted in many crowdsourcing approaches [11, 32, 7]. For ease of presentation, this paper considers the simple majority voting where k is an odd number. If more than or equal

Table 2: Notations.

$\mathcal{T}$	a set of microtasks for crowdsourcing
$\mathcal{W}$	a set of active workers working on $\mathcal{T}$
$\langle t_i, w\rangle$	assignment of microtask t_i and worker w
$\mathcal{W}_{t_i}$	a set of workers completing microtask t_i
$\mathbf{p}^w = \{p_i^w\}$	estimated accuracies of w
$\mathbf{q}^w = \{q_i^w\}$	observed accuracies of w
k	assignment size per microtask

to $\frac{k+1}{2}$ workers vote for a same answer (e.g., `YES`), then we take this consensus answer as the result of t_i, and consider t_i is *globally* completed. We can compute the accuracy of worker set $\mathcal{W}_{t_i}$, denoted by $Pr(\mathcal{W}_{t_i})$, based on the accuracy of each individual in $\mathcal{W}_{t_i}$. Specifically, this accuracy is the probability that more than half of the workers in $\mathcal{W}_{t_i}$ can provide the correct answer. Assuming the accuracies across workers are independent, we compute $Pr(\mathcal{W}_{t_i})$ as below.

$$Pr(\mathcal{W}_{t_i}) = \sum_{x=\frac{k+1}{2}}^{k} \sum_{\mathcal{W}_{t_i}^x} \Big(\prod_{w \in \mathcal{W}_{t_i}^x} p_i^w \prod_{w \in \mathcal{W}_{t_i} - \mathcal{W}_{t_i}^x} (1 - p_i^w) \Big) \quad (1)$$

where $\mathcal{W}_{t_i}^x$ is any x-size subsets of $\mathcal{W}_{t_i}$.

Based on worker set accuracy, an *optimal microtask assignment* problem assigns each microtask $t_i \in \mathcal{T}$ to a subset of workers $\mathcal{W}_{t_i}$ to maximize the sum of the probabilities that microtasks are correctly answered, i.e., $\max \sum_{t_i \in \mathcal{T}} Pr(\mathcal{W}_{t_i})$.

However, crowdsourcing has brought the following two great challenges that make microtask assignment much more difficult. First, due to the openness of crowdsourcing, we have limited, if not none, knowledge of workers. Thus, we cannot obtain the accuracy p_i^w upfront, which makes the computation of $Pr(\mathcal{W}_{t_i})$ in Equation (1) rather difficult. Second, the worker set $\mathcal{W}$ is dynamic: any existing worker may become inactive and new workers may become active. These challenges engender us to study the problem of *adaptive crowdsourcing*, which is described as follows.

Adaptive Crowdsourcing. This problem adaptively assigns microtasks to a dynamic set $\mathcal{W}$ of workers with unknown accuracy p_i^w to optimize the overall accuracy of crowdsourcing, i.e., $\max \sum_{t_i \in \mathcal{T}} Pr(\mathcal{W}_{t_i})$. Specifically, it on-the-fly estimates worker accuracy p_i^w based on the globally completed microtasks $\mathcal{T}^d \subseteq \mathcal{T}$. Then, based on the estimation, it adaptively assigns uncompleted microtasks in $\mathcal{T} - \mathcal{T}^d$ in iterations until all microtasks in $\mathcal{T}$ are globally completed, where each iteration is triggered by a request from an *active* worker w from a dynamic worker set $\mathcal{W}$.

For example, consider the microtasks in Table 1 and a worker set $\mathcal{W} = \{w_1, w_2, \ldots, w_5\}$. Adaptive crowdsourcing interacts with the workers in $\mathcal{W}$ for microtask assignment, where each assignment is triggered by a request from an active worker, say w_1. In the assignment, it considers the globally completed microtasks of w_1 to estimate the accuracy vector $\mathbf{p}^{w_1}$ of w_1, and assigns to w_1 a microtask with higher accuracy. After w_1 submits its answer to the assigned microtask, it adjusts the estimation of accuracies $\mathbf{p}^{w_1}$ based on the new $\mathcal{T}^d$, and continues to assign more microtasks.

For ease of presentation, we summarize the notations in this paper in Table 2, where some will be introduced later.

2.2 *i*Crowd Framework

Our *i*`Crowd` framework is illustrated in Figure 1. We develop *i*`Crowd` as an adaptive crowdsourcing component within the CDAS (Crowdsourcing Data Analytics System) project, along the lines of [22, 10]. The framework takes as input a set $\mathcal{T}$ of microtasks, and on-the-fly communicates with

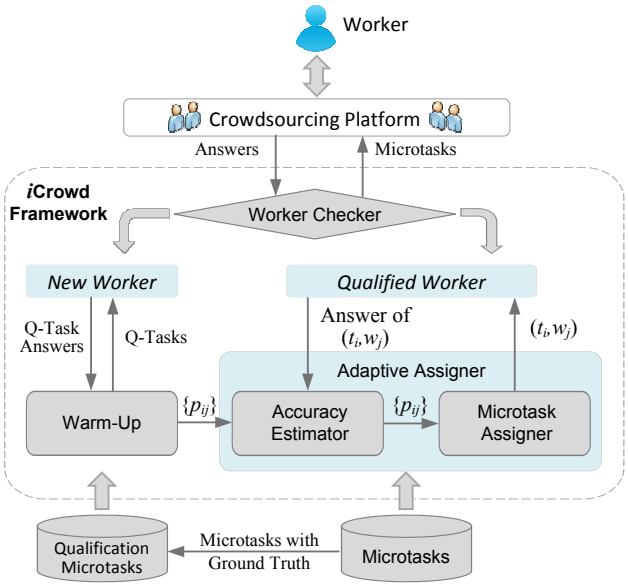


Figure 1: Framework of *i*`Crowd`.

a crowdsourcing platform (e.g., Amazon Mechanical Turk) for assigning microtasks to workers[1]. It adaptively assigns microtasks to workers by using the following components.

Adaptive Assigner. This component interacts with a requesting worker, and employs two modules, namely Accuracy Estimator and Microtask Assigner for adaptive microtask assignment.

Accuracy Estimator takes as input the globally completed microtasks $\mathcal{T}^d$ and estimates accuracies $\mathbf{p}^w$ for each worker w. Compared with existing approaches [22, 18, 30] that employ some qualification tasks with ground truth to compute an average accuracy for all the microtasks, the highlight of our estimation is to consider accuracy diversity across microtasks. The basic idea is to infer the accuracy of a worker's answer on a microtask based on her performance on the globally completed microtasks which are similar to the microtask. Specifically, as illustrated in Figure 2, suppose worker w has provided correct answer to t_1 about iPhone, and incorrect answer to t_2 about iPod. Then, Accuracy Estimator would have higher confidence on w's answers of other microtasks about iPhone (similar to t_1), while it would doubt her answers about iPod. To formulate this intuition, we introduce probabilistic models to evaluate the performance of w on globally completed microtasks $\mathcal{T}^d$, and employ a graph-based approach to infer accuracies $\mathbf{p}^w$ of w on the unassigned microtasks (see Section 3 for details).

Microtask Assigner takes as input the active workers with their accuracies estimated by Accuracy Estimator, and assigns a microtask to a requesting worker. It considers the dynamically estimated accuracies, and adaptively assigns a microtask to the top workers of the microtask with highest accuracies based on the current estimation. It also considers assigning the microtasks that would be more beneficial to improve the estimation of worker accuracies. Moreover, it devises effective indexes and develops efficient algorithms for supporting real-time microtask assignment. More details of microtask assigner can be found in Section 4.

[1]Note that the implementation details of the *online* communication between *i*`Crowd` and the crowdsourcing platform can be referred to Appendix A.

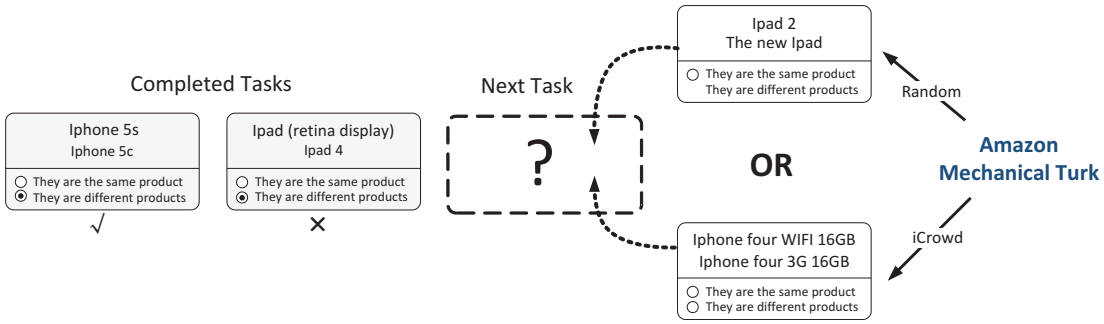


Figure 2: An Example of Using *i*Crowd to Assign Microtasks to Workers on Amazon Mechanical Turk.

Warm-Up. This component is used to address the cold-start problem: for a new worker w who does not vote for any globally completed microtask, *i*Crowd cannot estimate the worker accuracies $\mathbf{p}^w$. The component addresses the problem by leveraging some microtasks with ground truth, called qualification microtasks, for initial accuracy estimation. In the pre-processing step, it selects a small amount of "representative" microtasks from $\mathcal{T}$ and asks the requester to label ground truth for them. Then, when a new worker w requests for microtasks, it assigns some qualification microtasks to her (note that worker w is not aware that she is working on qualification microtasks). After w submits answers of the microtasks, the component compares her answers with the ground truth to estimate accuracies. Besides, it also seeks to eliminate "bad" workers based on qualification microtasks: if the average accuracy is lower than a given threshold, it rejects worker w as she is not qualified in working on the microtasks. For instance, given a threshold 0.6, if worker w correctly answers less than 3 qualification microtasks when completing 5 qualification microtasks, *i*Crowd rejects the worker (see Section 5 for more details).

3. ACCURACY ESTIMATION

This section introduces our accuracy estimation method that tackles accuracy diversity of workers on different microtasks. The basic idea of our method is to consider the "similarity" of microtasks: as observed from our empirical investigation in Section 6, the accuracies of a worker, although may vary in different domains, tend to be comparable on the microtasks in similar domains. As such, if we have observed that a worker w correctly completes some microtasks, we can infer that she would also perform well on microtasks in similar domains. Consider our example microtasks in Table 1: as t_1 and t_4 are similar (both of them are about "iPhone"), if we have observed that worker w correctly answers t_1, we can conjecture that w stands a better chance to correctly answer t_4 compared with other microtasks.

We formulate the aforementioned similarities of microtasks as a graph model, called *microtask similarity graph* (or similarity graph for simplicity). Formally, a similarity graph is a weighted undirected graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ where $\mathcal{T}$ is a set of microtasks, and an edge e_{ij} between microtasks t_i and t_j represents that t_i and t_j are similar and their similarity is denoted by s_{ij}. Figure 3 illustrates a similarity graph of our example microtasks, where the number in an edge represents the similarity score of the connected microtasks (we will explain how the number is computed in Section 3.3).

Based on the similarity graph, we estimate the accuracy vector $\mathbf{p}^w = \{p_1^w, p_2^w, \ldots, p_{|\mathcal{T}|}^w\}$ of worker w on the set $\mathcal{T}$ of

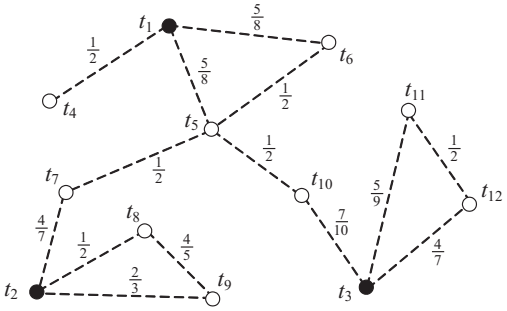


Figure 3: Similarity Graph of Example Microtasks.

Worker	Qualification	Completed Microtasks
w_1	t_1 √ t_2 × t_3 ×	(t_6, *YES*), (t_5, *NO*)
w_2	t_1 × t_2 √ t_3 ×	(t_6, *NO*), (t_7, *NO*)
w_3	t_1 × t_2 × t_3 √	(t_7, *YES*), (t_{11}, *YES*)
w_4	t_1 √ t_2 √ t_3 ×	(t_7, *NO*), (t_5, *YES*)
w_5	t_1 √ t_2 × t_3 √	(t_6, *YES*), (t_{11}, *NO*)

Figure 4: Completed Microtasks of Workers.

microtasks from the globally completed microtasks $\mathcal{T}^d \subseteq \mathcal{T}$. More formally, we introduce *observed accuracy*, denoted by q_i^w, to represent the probability that the answer of w on a globally completed microtask $t_i \in \mathcal{T}^d$ is correct. Similar to $\mathbf{p}^w$, for ease of presentation, we use a vector $\mathbf{q}^w$ to denote the observed accuracies of w on the microtask set $\mathcal{T}$. We will explain how to derive $\mathbf{q}^w$ from $\mathcal{T}^d$ in Section 3.2.

Now we are ready to formalize the problem of similarity-based accuracy estimation, which is defined as below.

DEFINITION 2 (SIMILARITY-BASED ACCURACY ESTIMATION). *Given a similarity graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ and observed accuracy vector $\mathbf{q}^w$ of worker w, it estimates accuracies $\mathbf{p}^w = \{p_1^w, p_2^w, \ldots, p_{|\mathcal{T}|}^w\}$ for unassigned microtasks in $\mathcal{T} - \mathcal{T}^d$.*

Consider our example similarity graph mentioned above and five workers shown in Figure 4. Given a set of globally completed microtasks, including the qualification microtasks (t_1, t_2 and t_3 in Figure 4) and the microtasks that reach consensus (e.g., t_6 given assignment size $k = 3$), and a worker w_1, the accuracy estimation problem wants to compute the accuracies of w_1 on the microtasks which are not assigned to w_1 yet, e.g., t_4, t_7, etc.

In the remaining part, we first introduce our model for similarity-based accuracy estimation in Section 3.1. Then, we discuss how to derive $\mathbf{q}^w$ from $\mathcal{T}^d$ in Section 3.2 and how to construct the similarity graph in Section 3.3.

3.1 Graph-Based Estimation Model

The basic idea of our graph-based estimation model is to estimate accuracies based on similarity between microtasks in the graph: if the observed accuracies in $\mathbf{q}^w$ are known on some globally completed microtasks, we can conjecture similar accuracies of the unassigned microtasks which are connected to the completed ones in the graph. Take the similarity graph in Figure 3 as an example. Given previous performance of worker w, say correctly answering t_2 and providing incorrect answer to t_3, our method would conjecture that w would have higher accuracies on the microtasks closely connected to t_2 (e.g., t_7, t_8 and t_9), while may not be accurate on those connected to t_3 (e.g., t_{10}, t_{11} and t_{12}).

However, it is non-trivial to exploit the similarity graph for accuracy estimation, as the estimated accuracies $\mathbf{p}^w$ need to satisfy not only the *local* similarity that each pair of connected microtasks have similar accuracies, but also the *global* similarity that microtasks in coherent sub-graphs have similar accuracies. Moreover, $\mathbf{p}^w$ also needs to be consistent with the observed accuracies in $\mathbf{q}^w$: estimated accuracies should not largely deviate from the observed accuracies. We formalize these two objectives as follows. For ease of presentation, we respectively use $\mathbf{p}$, $\mathbf{q}$, p_i and q_i to represent $\mathbf{p}^w$, $\mathbf{q}^w$, p_i^w and q_i^w in the remaining part, if there is no ambiguity.

1) We want to minimize accuracy differences between similar microtasks. Formally, recall that $S = \{s_{ij}\}$ denotes the similarity matrix, where s_{ij} is the similarity between two microtasks t_i and t_j. We introduce $(p_i - p_j)^2$ to measure the difference between the estimated accuracies of worker w on t_i and t_j. We choose this quadratic expression due to the following reasons. First, the lower the difference of the accuracies on t_i and t_j is, the smaller $(p_i - p_j)^2$ is, and vice versa. Thus, we can minimize accuracy differences between similar microtasks by minimizing $(p_i - p_j)^2$. Second, it will enable us to easily obtain the optimal solution of our optimization problem, which will be elaborated later in Lemma 1. Given the difference expression, this objective aims to minimize $(p_i - p_j)^2$ for the pair of microtasks with large similarity s_{ij}. We further consider to normalize the similarities. Let $S' = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$, where D denotes the diagonal matrix with the (i,i) element equal to the sum of i-th row of S, i.e., $D_{ii} = \sum_{j=1}^{n} s_{ij}$, where n is the number of microtasks in $\mathcal{G}$. Then, we formalize the overall accuracy difference between similar microtasks as $\frac{1}{2}\sum_{i,j}(\frac{s_{ij}}{\sqrt{D_{ii}}}p_i - \frac{s_{ij}}{\sqrt{D_{jj}}}p_j)^2$, which is the summation of accuracy differences between microtask pairs weighted by the normalized similarity.

2) We want to minimize accuracy difference between $\mathbf{p}$ and $\mathbf{q}$, as we do not want the estimate to deviate too much from the observed accuracies. We utilize $\frac{1}{2}\sum_{i=1}^{n}(p_i - q_i)^2$ to model the differences between $\mathbf{p}$ and $\mathbf{q}$. Note that we will discuss how to compute q_i in Section 3.2.

By considering the above factors, we compute accuracy vector $\mathbf{p}^*$ by solving the following optimization problem.

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \frac{1}{2}\Big(\sum_{i,j} s_{ij}\big(\frac{p_i}{\sqrt{D_{ii}}} - \frac{p_j}{\sqrt{D_{jj}}}\big)^2 + \alpha\sum_{i}^{n}(p_i - q_i)^2\Big), \tag{2}$$

where α is a parameter balancing the two factors: a larger α indicates that we prefer $\mathbf{p}$ will be more similar to $\mathbf{q}$, while a smaller α indicates that we are more concerned that the estimated accuracies should be closer on similar microtasks.

Take our example similarity graph in Figure 3 as an example. Suppose that a worker w has correctly answered t_1 and provided wrong answers to t_2 and t_3. Then, we have a vector $\mathbf{q}$ of observed accuracies containing $q_1 = 1$, $q_2 = 0$ and $q_3 = 0$ (we will explain the details of computing these observed accuracies in Section 3.2). Then, our optimization objective in Equation (2) aims to predict estimated accuracy p_i for microtask t_i such that the microtasks, which are closely connected in the similarity graph (e.g., the subgraph containing t_2 in the bottom right), have similar values in p_i, and these values are also similar to $\mathbf{q}$. For instance, given the above $\mathbf{q}$, the predicted $\mathbf{p}$ should be that the microtasks in the subgroup around t_1 have larger estimated accuracies than those in the subgraphs around t_2 and t_3.

The optimization problem can be analytically solved, as illustrated in the following lemma.

LEMMA 1. *The solution of Equation (2) is*

$$\mathbf{p}^* = \frac{\alpha}{1+\alpha}\cdot(I - \frac{1}{1+\alpha}S')^{-1}\mathbf{q}. \tag{3}$$

PROOF. We prove the lemma by differentiating the Equation (2) with respect to $\mathbf{p}$ and then compute $\mathbf{p}^*$ by taking the first-order derivative equal to zero. □

As it is challenging to directly compute the inverse matrix $(I - \frac{1}{1+\alpha}S')^{-1}$, we employ the personalized pagerank algorithm [14] to iteratively compute $\mathbf{p}$. Specifically, we set vector $\mathbf{p}$ as the observed one $\mathbf{q}$ initially, and then update $\mathbf{p}$ by using the following equation.

$$\mathbf{p} = \frac{1}{1+\alpha}\cdot\mathbf{p}S' + \frac{\alpha}{1+\alpha}\cdot\mathbf{q}. \tag{4}$$

As illustrated in Equation (4), in each iteration, each element $p_i \in \mathbf{p}$ is computed by the sum of it nearby accuracies with similarity as weight at probability $\frac{1}{1+\alpha}$, or set to its observed accuracy q_i at probability $\frac{\alpha}{1+\alpha}$.

LEMMA 2. *The iterative algorithm based on Equation (4) can compute the optimal solution $\mathbf{p}^*$ in Equation (3).*

PROOF. This lemma is proved in [35]. □

The iterative algorithm based on Equation (4) is also very expensive, and it is not applicable for online accuracy estimation. To address this problem, we propose an efficient algorithm by utilizing a nice *linearity* property of personalized pagerank. Specifically, let $\mathbf{q}_{t_i}$ denote the observed accuracy vector that only q_i for microtask t_i is 1 and others are 0. Let $\mathbf{p}_{t_i}$ denote the converged result of Equation (4) given $\mathbf{q}_{t_i}$. There is a good property that we can compute $\mathbf{p}^*$ by a weighted summation of $\mathbf{p}_{t_i}$, as shown below.

LEMMA 3 (LINEARITY [14]). *Given $\mathbf{p}_{t_i}$ for each microtask t_i and $\mathbf{q}$, the optimal estimated accuracy vector $\mathbf{p}^*$ can be computed as $\mathbf{p}^* = \sum_{q_i \in \mathbf{q}} q_i \cdot \mathbf{p}_{t_i}$.*

Based on Lemma 3, we develop an efficient online estimation algorithm as shown in Algorithm 1. The algorithm takes as input a worker w, microtask set $\mathcal{T}$, and the set of globally completed microtasks $\mathcal{T}^d$. In particular, we also consider the qualification microtasks with ground truth as globally completed. The algorithm estimates accuracy vector $\mathbf{p}$ of worker w through the following steps. In the

Algorithm 1: EstimateAccuracy (w, $\mathcal{T}$, $\mathcal{T}^d$)

Input: w: A worker; $\mathcal{T}$: A microtask set ;
$\mathcal{T}^d$: A set of globally completed microtasks
Output: $\mathbf{p}^w$: Estimated accuracies of w

```
1 begin
    /* Offline Computation                          */
2   Generate a similarity graph G on T ;
3   for Each microtask t_i in G do
4     Compute p_{t_i} iteratively via Equation (4) ;
    /* Online Estimation                            */
5   q^w ← ComputeObserved (T^d) ;
6   p^w ← Σ_{q_i^w ∈ q^w} q_i^w · p_{t_i} ;
7   return p^w ;
8 end
```

offline component, it generates a similarity graph $\mathcal{G}$ by computing microtask similarities, and pre-computes a vector $\mathbf{p}_{t_i}$ for each microtask t_i in $\mathcal{G}$ via Equation (4). For online estimation, it first computes observed accuracies $\mathbf{q}$ based on completed microtasks $\mathcal{T}^d$ by calling function ComputeObserved (see Section 3.2), and then employs the linearity property to compute $\mathbf{p}$. Thanks to the linearity property, the algorithm is efficient with time complexity $\mathcal{O}(|\mathcal{T}|)$.

For example, consider estimating accuracies for worker w_1 in Figure 4. In offline, the algorithm computes $\mathbf{p}_{t_i}$ for each t_i based on the graph in Figure 3. For instance, to compute $\mathbf{p}_{t_1}$, it initially sets $\mathbf{p}_{t_1} = \mathbf{q}_{t_1} = [1, 0, \ldots, 0]$, and then updates $\mathbf{p}_{t_1}$ via Equation (4) to obtain a converged vector. In the online component, after computing $\mathbf{q}^{w_1}$, it computes accuracies of w_1 as $\mathbf{p}^{w_1} = \sum_{q_i^{w_1} \in \mathbf{q}^{w_1}} q_i^{w_1} \cdot \mathbf{p}_{t_i}$.

3.2 Observed Accuracy Estimation

In this section, we discuss how to estimate the observed accuracies $\mathbf{q}^w$ that model worker w's previous performance on the globally completed microtasks.

The estimation is trivial if t_i is a qualification microtask with ground truth. In this case, we compare the answer from w with the ground truth of t_i, and set $q_i^w = 1$ if they are the same, or $q_i^w = 0$ otherwise. The non-trivial case is when t_i does not have ground truth. In this case, we estimate q_i^w by comparing w's answer with the consensus answer of microtask t_i. The basic idea is that, if the two answers are the same, we can use the probability that the consensus answer is correct to estimate q_i^w. Otherwise, we consider the probability of the complement event, i.e., the consensus answer is incorrect. Formally, we respectively use $\mathcal{W}_1$ and $\mathcal{W}_2$ to denote the workers of t_i having the answer equal to and not equal to the consensus answer. Then, the observed accuracy of worker w can be computed as

$$q_i^w = \begin{cases} \frac{P_1\overline{P}_2}{P_1\overline{P}_2 + \overline{P}_1 P_2}, & if\ \mathtt{ans}_i^w = \mathtt{ans}_i^* \\ \frac{\overline{P}_1 P_2}{P_1\overline{P}_2 + \overline{P}_1 P_2}, & if\ \mathtt{ans}_i^w \neq \mathtt{ans}_i^* \end{cases} \quad (5)$$

where $\mathtt{ans}_i^w$ is w's answer, $\mathtt{ans}_i^*$ is the consensus answer, $P_1 = \prod_{w' \in \mathcal{W}_1} p_i^{w'}$, $\overline{P}_1 = \prod_{w' \in \mathcal{W}_1} 1 - p_i^{w'}$, $P_2 = \prod_{w' \in \mathcal{W}_2} p_i^{w'}$ and $\overline{P}_2 = \prod_{w' \in \mathcal{W}_2} 1 - p_i^{w'}$. Note that the above $p_i^{w'}$ can be estimated by the previously estimated accuracy of w'. When estimating $\mathbf{q}^w$ for the first time, we use the average accuracy returned by the Warm-Up component as an estimate.

Figure 4 provides some completed microtasks of the five workers where $t^{\checkmark}$ ($t^{\times}$) represents qualification t is correctly (incorrectly) answered. For worker w_1, we obtain $q_1^{w_1} = 1$, $q_2^{w_1} = q_3^{w_1} = 0$ based on the qualification. We compute $q_6^{w_1}$ for a globally completed microtask t_6 (assignment size $k = 3$). We consider the workers of t_6, $\{w_1, w_2, w_5\}$, and find w_1's answer being equal to the consensus answer. Thus, we compute $q_6^{w_1} = \frac{p_6^{w_1} p_6^{w_5} (1 - p_6^{w_2})}{p_6^{w_1} p_6^{w_5} (1 - p_6^{w_2}) + (1 - p_6^{w_1})(1 - p_6^{w_5}) p_6^{w_2}}$.

3.3 Discussion on Microtask Similarity

In this section, we discuss how to derive microtask similarities for generating the similarity graph. We utilize the well-known similarity techniques and examine the effect of these techniques in the experiments (see Appendix D.1). Note that this paper focuses on leveraging similarity for better accuracy estimation, and leaves a thorough study on similarity metrics as future work.

1) For textual microtasks, e.g., our example in Table 1, we can employ existing similarity metrics, e.g., Jaccard similarity, Edit distance, etc, to compute the similarity between microtasks. For instance, we derive our example similarity graph shown in Figure 3 by using Jaccard similarity. We first take each of microtasks as a set of tokens, and compute Jaccard similarity as the ratio of their intersection size to their union size, and consider that two microtasks are similar if their Jaccard similarity is not smaller than a given threshold. For example, the edge between microtasks t_2 and t_7 represents their similarity is 4/7, which is computed by Jaccard similarity between their token sets in Table 1. We also set a similarity threshold 0.5 to neglect the microtask pairs with similarity lower than 0.5. We can employ the techniques in [33] to select appropriate similarity metrics and thresholds. In addition, we can also exploit existing topic models, e.g., Latent Dirichlet Allocation (LDA) [6] to obtain a topic distribution for each microtask, and compute microtask similarity based on the topic distributions. In our experiment, we utilized the LDA-based similarity and achieved good performance.

2) For microtasks that can be represented as multiple-dimension features, e.g., images, spatial objects, etc, we can use the Euclidean similarity to quantify their similarity. For instance, consider microtasks for verifying place names for points-of-interest (POIs) in the map. Given two microtasks t_i and t_j, we consider their corresponding POIs (x_1, x_2) and (y_1, y_2). Then, we compute their Euclidean distance $\mathtt{dist}(t_i, t_j) = \sqrt{\sum_{d=1}^{2} (x_d - y_d)^2}$, and use $1 - \frac{\mathtt{dist}(t_i, t_j)}{\tau_d}$ to normalize their similarity between 0 and 1, where τ_d is the maximum distance between POIs in the microtasks.

3) For some more complicated microtasks, e.g. verifying translations of a paragraph of text, we can use classification-based methods to derive their similarity. We can train a classifier based on a training set. Then for each pair of microtasks, if we classify them as a similar pair, their similarity is 1, and 0 otherwise. We may use existing classification algorithms, e.g. SVM, to find similar microtasks.

4. ADAPTIVE TASK ASSIGNMENT

This section presents our adaptive approach for microtask assignments based on estimated accuracies. We introduce an adaptive assignment framework in Section 4.1 and discuss the *optimal microtask assignment* problem in Section 4.2.

4.1 Adaptive Assignment Framework

The basic idea of our microtask assignment framework is to adaptively "seek" for the *top workers* with higher accura-

cies to complete microtasks, as the estimated accuracies are continuously updated. A high-level overview of our framework is illustrated in Figure 5. At each time, the framework aims to assign the microtasks, which are not globally completed, to the current set of active workers. It first adaptively finds the top workers having the highest estimated accuracy p_i^w to complete each unassigned microtask t_i based on the up-to-date accuracy estimation, as illustrated by the dotted arrow in the figure. As worker w may be taken as top workers by multiple microtasks, the framework then determines a microtask to which w can contribute the most and assigns this microtask to w (solid arrows). Moreover, when a worker submits her answer to the assigned microtask, the framework adaptively updates the assignment scheme based on the new accuracy estimation.

For example, consider the five workers $w_1, w_2, \ldots, w_5$ in Figure 5. The framework considers the up-to-date accuracies $p_1^{w_1}, p_2^{w_2}, \ldots$ estimated by Accuracy Estimator and selects top workers, e.g., the top workers of t_2 are w_1, w_3 and w_4. As some worker (e.g., w_3) may be selected as top worker for multiple microtasks (e.g., t_1 and t_2), the framework then determines assigning which microtask is more "beneficial" (will be explained later). Finally, our framework generates a scheme that assigns t_1 to workers w_2 and w_5, and t_2 to workers w_1, w_3 and w_4.

To formalize the above idea, our framework utilizes the following three steps for adaptive microtasks assignment.

Step 1 - Top worker set computation. This step first identifies *active* workers which are ready to work on our microtasks. We can use different methods for identifying active workers. One way is to use a time window (e.g., 30 minutes): if the span from the last time when worker w requests microtasks to the current time is smaller than the window, we consider w is active. Otherwise, w is inactive. Another method is to examine if or not a worker is still holding Human Intelligence Tasks (HITs) containing the microtasks (see Appendix A for more details): if w is holding a HIT, we consider w is active; otherwise, w is inactive.

Taking the set of active workers as candidates, the framework then computes for each microtask the top workers. Formally, let $\mathcal{W}^d(t_i)$ denote the set of workers who have been assigned to t_i, i.e., the workers either have completed or those are still working on t_i. Let $\mathcal{W}^u(t_i) = \mathcal{W} - \mathcal{W}^d(t_i)$ denote the workers to which t_i can be assigned. Recall that each microtask has an assignment size k that it could be assigned to k workers. We define the top workers as below.

Definition 3 (Top Worker Set). *Top worker set of a microtask t_i, denoted by $\widehat{\mathcal{W}}(t_i)$, is a set of top $k' = k - |\mathcal{W}^d(t_i)|$ workers with the highest accuracies on t_i, i.e., $\forall w \in \widehat{\mathcal{W}}(t_i), \forall w' \in \mathcal{W}^u(t_i) - \widehat{\mathcal{W}}(t_i), p_i^w \geq p_i^{w'}$.*

Table 3 provides some example top worker sets ($k = 3$). For instance, microtask t_4 has no assigned set $\mathcal{W}^d(t_i)$ and its top worker set consists of $\{w_5, w_4, w_1\}$ with the highest accuracies (the numbers in parentheses). In addition, as t_{11} has been assigned to worker w_2 in previous interaction, its top worker set only contains two workers, w_5 and w_3.

Step 2 - Optimal microtask assignment. This step takes the microtasks and their top worker sets as input and generates an assignment scheme that satisfies the following two properties: 1) each worker w could be assigned with at most *one* microtask at each time, and 2) the number of workers that a microtask t_i can be assigned to is not greater than the available assignment size of t_i, i.e., the $k' = k - |\mathcal{W}^d(t_i)|$ in Definition 3. Consider our example in Table 3: both t_4 and t_{11} take w_5 as one of their top workers. However, w_5 can only work on one microtask from $\{t_4, t_{11}\}$ before w_5 submits her answer to the assigned microtasks and continues to request more microtasks. Under the above constraints, we want to find an *optimal* assignment scheme that achieves the highest overall accuracy. We formalize the problem of optimal microtask assignment and develop an efficient algorithm to solve the problem, which will be described in Section 4.2.

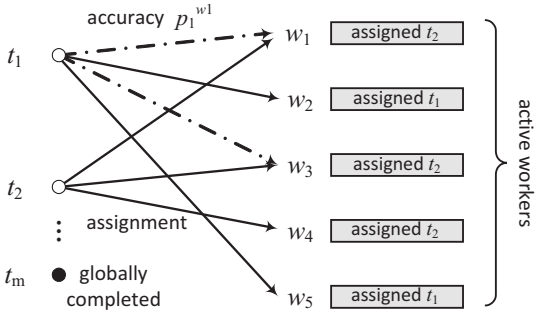


Figure 5: Overview of Microtask Assignment.

Table 3: An Example of Microtask Assignment.

Task	Top Worker Set $\widehat{\mathcal{W}}(t_i)$	Assigned $\mathcal{W}^d(t_i)$
t_4	$(w_5, 0.75), (w_4, 0.7), (w_1, 0.6)$	$\emptyset$
t_{11}	$(w_5, 0.85), (w_3, 0.8)$	$(w_2, 0.7)$
t_9	$(w_4, 0.85), (w_2, 0.75), (w_1, 0.7)$	$\emptyset$
t_{10}	$(w_3, 0.7), (w_1, 0.6)$	$(w_5, 0.8)$

Step 3 - Worker performance testing. After the previous step, some workers may still not be assigned with microtasks, as they are *currently* not taken as top workers of any microtask. This may be attributed to our limited knowledge about these workers. For example, suppose that a worker w has completed all the microtasks she is good at according to our accuracy estimation, say the microtasks in the subgraph near t_1 shown in Figure 3. Then, our Accuracy Estimator is not sure if w can perform well in the other two subgraphs near t_2 and t_3, as w only completed a limited number of microtasks in these subgraphs.

In this case, our assignment framework *actively* tests the performance of w by considering the following two factors. The first factor is the uncertainty of the estimated accuracy of w on the microtask: we prefer to test a microtask, on which Accuracy Estimator is not certain if w can perform well. We use the *estimation variance* to model the uncertainty. Suppose that w has completed N microtasks that are similar to t' (e.g., some microtasks in the left-bottom subgraph similar to t_2 in Figure 3), and N_1 microtasks are correctly answered (based on our estimation model in Section 3.2) while N_0 microtasks are incorrect. We estimate the estimation variance by assuming N_1 and N_0 follow a beta distribution (which is a commonly-used assumption in estimation) $\beta(N_1 + 1, N_0 + 1)$, and estimate the uncertainty by the variance of the beta distribution, i.e., $\frac{(N_1+1)\cdot(N_0+1)}{(N_1+N_0+2)^2\cdot(N_1+N_0+3)}$. Then, we prefer to select the microtask with larger variance. The second factor is the accuracies of existing workers which have been assigned to t': we prefer the microtask with high accuracies to make the performance

Algorithm 2: AssignTask ($\mathcal{W}, \mathcal{T}, \mathcal{T}^d, \{\mathbf{p}^w\}$)

Input: $\mathcal{W}$: Active workers; $\mathcal{T}$: microtask set; $\mathcal{T}$: globally completed microtask set; $\{\mathbf{p}^w\}$: Estimated accuracies of workers
Output: Asg $= \{\langle w_j, t_i\rangle\}$: Assignments

begin
  // Step 1: Generating top worker sets ;
  for *each microtask* $t_i \in \mathcal{T} - \mathcal{T}^d$ **do**
    $\widehat{\mathcal{W}}(t_i) \leftarrow$ GenTopWorkers (t_i, $\{\mathbf{p}^w\}$) ;
  // Step 2: Compute optimal microtask assignment ;
  $\mathcal{A}^* \leftarrow$ OptAssign ($\mathcal{T} - \mathcal{T}^d, \mathcal{W}, \{\widehat{\mathcal{W}}(t_i)\}$) ;
  for *each* $\langle t, \mathcal{W}_t\rangle \in \mathcal{A}^*$ **do**
    Insert $\langle w, t\rangle$ into Asg for any $w \in \mathcal{W}_t$;
  // Step 3: Test performance for workers ;
  for *each* w *in* $\mathcal{W}$ *without assignment* **do**
    $t' \leftarrow$ ComputeTestAssignment (w) ;
    Insert $\langle w, t'\rangle$ into Asg ;
  Return Asg ;
end

test more reliable. We use the estimated accuracies to measure quality of the existing worker set.

Algorithm 2 summarizes the pseudo-code of our adaptive assignment framework. It first generates top worker sets for the microtasks that are not globally completed. Then, it employs the top worker sets to compute an optimal assignment scheme. For the workers who are not assigned with microtasks, it further tests their accuracies by assigning microtasks that have been assigned to other workers. We next analyze the time complexity. The complexity of the first step is $\mathcal{O}(|\mathcal{T}| \cdot |\mathcal{W}|)$, as it computes top workers for each microtask. We will prove optimal microtask assignment OptAssign in the second step is NP-hard and devise a greedy approximate algorithm with complexity $\mathcal{O}(|\mathcal{T}|^2)$ in Section 4.2. The complexity of the third step is $\mathcal{O}(|\mathcal{W}| \cdot |\mathcal{T}|)$.

4.2 Optimal Microtask Assignment

This section formalizes the *optimal microtask assignment* problem mentioned in Section 4.1 and provides an algorithm to solve the problem. The basic idea is to find an assignment scheme to make as many microtasks to be globally completed as possible, so as to immediately obtain the consensus result of these microtasks and improve accuracy estimation. Note that, to globally complete a microtask t_i, we can assign the top worker set $\widehat{\mathcal{W}}(t_i)$ to t_i (see Definition 3). For example, consider the microtasks shown in Table 3. Suppose that we assign t_{11} to its top worker set $\widehat{\mathcal{W}}(t_{11}) = \{w_5, w_3\}$. This microtask can be globally completed after the workers submit their answers and we can then employ Algorithm 1 to improve accuracy estimation of these workers. However, if two microtasks t_i and t_j have common top worker sets, these two microtasks cannot be completed at the same time, as a worker can only complete one microtask at a time. For instance, if we assign t_4 to its top worker set $\widehat{\mathcal{W}}(t_4) = \{w_5, w_4, w_1\}$, w_5 and w_1 cannot respectively be assigned with t_{11} and t_9 until they submit answers to t_4. Under this constraint, we prefer to find a scheme that assigns workers to the microtasks with higher overall accuracy and make these microtasks globally completed. To formalize this intuition, we introduce the optimal microtask assignment problem as follows.

Definition 4 (Optimal Task Assignment). *Given a set of uncompleted microtasks $\mathcal{T} - \mathcal{T}^d$, a set of active workers $\mathcal{W}$, and top worker set $\widehat{\mathcal{W}}(t_i)$ for each microtask t_i, it finds an optimal assignment scheme $\mathcal{A}^* = \{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle\}$ to maximize the summation of the overall worker accuracy of each microtask in the scheme, i.e.,*

$$\mathcal{A}^* = \arg\max_{\mathcal{A}} \sum_{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle \in \mathcal{A}} \Big(\sum_{w \in \widehat{\mathcal{W}}(t_i)} p_i^w \Big)$$
$$\texttt{s.t.}\ \forall t_i, t_j, \widehat{\mathcal{W}}(t_i) \cap \widehat{\mathcal{W}}(t_j) = \emptyset.$$

where $\sum_{w \in \widehat{\mathcal{W}}(t_i)} p_i^w$ is the overall accuracy of workers that globally complete t_i. Summing such overall accuracy of each microtask in $\mathcal{A}$[2], $\sum_{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle \in \mathcal{A}} \big(\sum_{w \in \widehat{\mathcal{W}}(t_i)} p_i^w \big)$ can measure the accuracy of all globally completed microtasks: maximizing this formula can make as many microtasks to be globally completed with high-accuracy workers as possible.

For example, an assignment scheme in Table 3 is $\mathcal{A} = \{\langle t_9, \widehat{\mathcal{W}}(t_9)\rangle, \langle t_{11}, \widehat{\mathcal{W}}(t_{11})\rangle\}$, which makes t_9 and t_{11} to be globally completed by respectively assigning top worker sets $\widehat{\mathcal{W}}(t_9)$ and $\widehat{\mathcal{W}}(t_{11})$. The overall accuracy of this assignment scheme is $\sum_{w \in \widehat{\mathcal{W}}(t_{11})} p_{11}^w + \sum_{w \in \widehat{\mathcal{W}}(t_9)} p_9^w$.

Next, we show the optimal microtask assignment problem is NP-hard, as shown in the following lemma (see Appendix B for the proof).

Lemma 4. *The problem of optimal microtask assignment is NP-hard.*

A Greedy approximation algorithm. As the problem of microtask selection is NP-hard, we develop a greedy-based approximation algorithm to efficiently solve it, as shown in Algorithm 3. It takes as input the microtasks $\mathcal{T} - \mathcal{T}^d$ that are not globally completed, a set $\mathcal{W}$ of active workers and top worker set $\widehat{\mathcal{W}}(t_i)$ of each microtask, and produces an assignment scheme $\mathcal{A}^* = \{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle\}$. To this end, it first computes candidate assignments $\mathcal{A}^c$ by considering each $t_i \in \mathcal{T} - \mathcal{T}^d$ as well as its $\widehat{\mathcal{W}}(t_i)$, and then chooses assignments from $\mathcal{A}^c$ iteratively. In each iteration, it chooses the assignment $\langle t, \widehat{\mathcal{W}}(t)\rangle$ with the maximum average worker accuracy $\sum_{w \in \widehat{\mathcal{W}}(t)} p_t^w / |\widehat{\mathcal{W}}(t)|$. After inserting $\langle t, \widehat{\mathcal{W}}(t)\rangle$ into $\mathcal{A}^*$, it eliminates the assignment $\langle t', \widehat{\mathcal{W}}(t')\rangle \in \mathcal{A}^c$ overlapping with $\widehat{\mathcal{W}}(t)$. Finally, if there is no more candidate in

[2]In some special cases, $\mathcal{A}$ may be possibly empty.

Algorithm 3: GreedyAssign ($\mathcal{T} - \mathcal{T}^d, \mathcal{W}, \{\widehat{\mathcal{W}}(t_i)\}$)

Input: $\mathcal{T} - \mathcal{T}^d$: uncompleted tasks; $\mathcal{W}$: Active workers; $\{\widehat{\mathcal{W}}(t_i)\}$: Top worker set of each task t_i;
Output: $\mathcal{A}^* = \{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle\}$: assignment results

begin
  $\mathcal{A}^c \leftarrow \emptyset$;
  for $t_i \in \mathcal{T} - \mathcal{T}^d$ **do** $\mathcal{A}^c \leftarrow \mathcal{A}^c \cup \{\langle t_i, \widehat{\mathcal{W}}(t_i)\rangle\}$;
  $\mathcal{A}^* \leftarrow \emptyset$;
  while $\mathcal{A}^c$ *is not empty* **do**
    Choose a $\langle t, \widehat{\mathcal{W}}(t)\rangle \in \mathcal{A}^c$ with the maximum $\frac{\sum_{w \in \widehat{\mathcal{W}}(t)} p_t^w}{|\widehat{\mathcal{W}}(t)|}$;
    $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{\langle t, \widehat{\mathcal{W}}(t)\rangle\}$;
    for *Each* $\langle t', \mathcal{W}_{t'}\rangle \in \mathcal{A}^c$ **do**
      if $\widehat{\mathcal{W}}(t') \cap \widehat{\mathcal{W}}(t) \neq \emptyset$ **then**
        Remove $\langle t', \mathcal{W}_{t'}\rangle$ from $\mathcal{A}^c$;
  return $\mathcal{A}^*$;
end

$\mathcal{A}^c$, it terminates and outputs the scheme $\mathcal{A}^*$. The time complexity of the greedy algorithm is $\mathcal{O}(|\mathcal{T}|^2)$.

Take the microtasks in Table 3 as an example. The greedy algorithm produces $\mathcal{A}^*$ for the microtasks as follows. In the first iteration, it computes $\frac{\sum_{w\in\widehat{\mathcal{W}}(t_i)} p_i^w}{|\widehat{\mathcal{W}}(t_i)|}$ for each microtask and chooses t_{11} with the highest score. Then, it inserts $\langle t_{11}, \{w_5, w_3\}\rangle$ into $\mathcal{A}^*$ and removes $\langle t_4, \widehat{\mathcal{W}}(t_4)\rangle$, $\langle t_{10}, \widehat{\mathcal{W}}(t_{10})\rangle$ from $\mathcal{A}^c$. Similarly, in the second iteration, the algorithm chooses $\langle t_9, \{w_4, w_2, w_1\}\rangle$ and inserts it into $\mathcal{A}^*$. Then, given an empty $\mathcal{A}^c$, the algorithm terminates and outputs $\mathcal{A}^*$.

5. QUALIFICATION TASK ASSIGNMENT

Our framework in Algorithm 2 may not dispatch microtask to a worker w if w is not contained in any top worker set. This will occur if 1) w is a new worker, or 2) w performs worse than others on all microtasks. In both cases, our framework needs to further test the quality of worker w by assigning her microtasks. We propose to assign qualification microtasks with ground truth to this end. Intuitively, in the former case, these microtasks are used to test which microtask w is good at. In the latter case, we have known that w is less competitive on the microtask similar to what she has completed. Thus, we need to test whether or not w is good at other microtasks, as mentioned before.

It is worth noting that the requester can only label ground truth to a limited number of qualification microtasks. Thus, we need to determine what microtasks have larger "benefit" to be qualification microtasks. We introduce a method for selecting qualification microtasks. The basic idea is to select the microtasks that have the maximum "influence" to other microtasks in the similarity graph. Intuitively, the influence captures how many microtasks we can infer that a worker has potential to provide correct answer if she correctly answers qualification microtasks. Take the similarity graph in Figure 3 as an example. If we can only select 3 microtasks, a good choice is $\{t_1, t_2, t_3\}$ as they can influence more microtasks related to iPhone, iPad and iPod. On the other hand, the microtasks $\{t_1, t_4, t_5\}$ only have strong influence on microtasks about iPhone and weak influence on other microtasks, and thus they can only help us test whether workers are good at this topic.

Based on the above intuition, consider qualification microtasks $\mathcal{T}^{\mathtt{Q}} \subseteq \mathcal{T}$. We model the influence of $\mathcal{T}^{\mathtt{Q}}$ as the number of non-zero values in the estimated accuracy vector inferred from the qualification microtasks in $\mathcal{T}^{\mathtt{Q}}$. Formally, based on the linearity property introduced in Lemma 3, the estimated accuracy vector can be computed by $\sum_{t_i\in\mathcal{T}^{\mathtt{Q}}} \mathbf{p}_{t_i}$ where $\mathbf{p}_{t_i}$ is the converged result of Equation (4) under the condition that the observed accuracies satisfy that only q_i for microtask t_i is 1 and others are 0. The influence is then computed by $\mathtt{INF}(\mathcal{T}^{\mathtt{Q}}) = \sum_{p_i\in\sum_{t_i\in\mathcal{T}^{\mathtt{Q}}}\mathbf{p}_{t_i}} \mathcal{I}(p_i \neq 0)$ where $\mathcal{I}(\cdot)$ is an indicator function (i.e., $\mathcal{I}(p_i \neq 0) = 1$ if $p_i \neq 0$, and $\mathcal{I}(p_i \neq 0) = 0$ otherwise). Now, we are ready to introduce the problem of qualification selection.

Definition 5 (Qualification Microtask Selection). *Given a similarity graph $\mathcal{G}$ and a number $\mathtt{Q}$, it selects a subset of microtasks $\mathcal{T}^{\mathtt{Q}} \subseteq \mathcal{T}$ satisfying: 1) the size $|\mathcal{T}^{\mathtt{Q}}| \leq \mathtt{Q}$, and 2) the influence of $\mathcal{T}^{\mathtt{Q}}$, $\mathtt{INF}(\mathcal{T}^{\mathtt{Q}})$ is maximized.*

Lemma 5. *Qualification microtask selection is NP-hard.*

Proof. See Appendix C for the proof. □

Algorithm 4: SelectQualification $(\mathcal{G}, \mathcal{T}, \mathtt{Q})$

Input: $\mathcal{G}$: A similarity graph; $\mathcal{T}$: microtask set; $\mathtt{Q}$: Number of qualification microtasks
Output: $\mathcal{T}^{\mathtt{Q}}$: Selected qualification microtasks

```
begin
    for Each microtask t_i in T do
        Compute p_{t_i} iteratively on G via Equation (4) ;
    T^Q ← ∅ ;
    for i = 1 ... Q do
        Find a microtask t* satisfying:
            t* = arg max_{t∈T−T^Q} {INF(T^Q ∪ {t}) − INF(T^Q)},
            where INF(T^Q) = Σ_{p_i∈Σ_{t_i∈T^Q} p_{t_i}} I(p_i ≠ 0) ;
        T^Q ← T^Q ∪ {t*} ;
    Return T^Q ;
end
```

We can devise a greedy algorithm for qualification microtask selection, as shown in Algorithm 4. The algorithm first computes $\mathbf{p}_{t_i}$ iteratively on the similarity graph $\mathcal{G}$ via Equation (4). Then, it selects qualification in $\mathtt{Q}$ iterations. In each iteration, it selects a task t^* that maximizes the marginal influence, i.e., $t^* = \arg\max_{t\in\mathcal{T}-\mathcal{T}^{\mathtt{Q}}} \{\mathtt{INF}(\mathcal{T}^{\mathtt{Q}} \cup \{t\}) - \mathtt{INF}(\mathcal{T}^{\mathtt{Q}})\}$.

We can prove that the approximation ratio of this greedy algorithm is $1 - \frac{1}{e}$ where e is the base of natural logarithm. Next, we analyze its time complexity. It has $\mathtt{Q}$ iterations, and, in each iteration, it computes $\mathtt{INF}(\mathcal{T}^{\mathtt{Q}} \cup \{t\})$ for each microtask $t \in \mathcal{T}$, which needs to scan all microtask in $\mathbf{p}_{t_i}$. Overall, the complexity of Algorithm 4 is $\mathcal{O}(\mathtt{Q} \cdot |\mathcal{T}|^2)$.

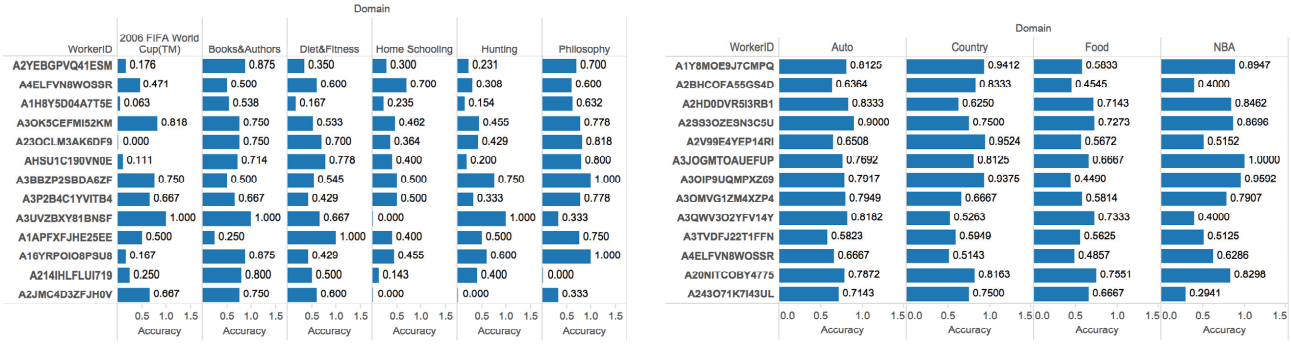
6. EXPERIMENTS

6.1 Experiment Setup

Datasets: We conducted experiments on real-world crowdsourcing platform Amazon Mechanical Turk (AMT) and evaluated the approaches on two real datasets.

1) `YahooQA` dataset: Evaluating quality of Question-Answers. The first dataset contained a set of microtasks that asked workers to evaluate whether a user-generated answer from *Yahoo Answers* can appropriately address its corresponding question. An example question and one of its answers were "*Who first proposed Heliocentrism?*" and "*Nicolaus Copernicus, a Renaissance mathematician and astronomer.*". A worker was required to do a Yes/No selection, i.e., whether the answer can appropriately address the question. We generated the ground truth of each task based on the ratings from Yahoo Answers: if an answer was rated as the "Best Answer" by the asker and attracted many more "Thumbs Up" than "Thumbs Down", the answer was labeled with 1. On the other hand, if the answer was not selected by the asker and had many more "Thumbs Down", it was labeled with 0. We selected 110 question-answer pairs as tasks in six domains, 2006 FIFA World Cup (`FF`), Books & Authors (`BA`), Diet & Fitness (`DF`), Home Schooling (`HS`), Hunting (`HT`) and Philosophy (`PH`). Table 4 shows statistics of `YahooQA`.

2) `ItemCompare` dataset: This dataset contained a set of microtasks that ask workers to compare two items based on a specified comparison criteria. We used four domains, `Food`, `NBA`, `Auto` and `Country`. In the `Food` domain, a microtask asked workers to compare two food items, say Chocolate and Honey, and identified which one had more calories. In the



(a) YahooQA dataset. (b) ItemCompare dataset.

Figure 6: Diverse Workers' Accuracies Across Different Domains (Here, we only list workers that completed more than 20 microtasks).

Table 4: Dataset Statistics.

Dataset	YahooQA	ItemCompare
# of microtasks	110	360
# of domains	6	4
# of workers	25	53

NBA domain, workers compared two NBA teams, say Los Angeles Lakers and Milwaukee Bucks, and verified which team won more NBA champions. In the Auto domain, workers compared two cars, say 2014 Toyota Camry and 2014 Lexus ES, and verified which one was more fuel efficient. In the Country domain, workers compared two countries, say Brazil and Canada, and verified which one had larger total area. We generated 360 microtasks where each domain had 90 microtasks, and manually labelled the ground truth.

We published all the microtasks on AMT and evaluated different approaches. We put 10 microtasks as a batch in a Human Intelligence Task (HIT), and set the price of each assignment as \$0.1. To ensure that all the approaches were compared on the same set of workers, we set the "Number of Assignments per HIT" to a large number (10 in our experiments) to collect enough answers. Based on the collected answers, we ran different approaches for task assignment and compared their performance. Table 4 shows statistics of the datasets and answers collected from AMT.

Baseline approaches: We implemented our methods and compared with baseline approaches. Note that we compared both task assignment and aggregation: Given an approach, we used its task assignment strategy to obtain answers from workers and employed its aggregation strategy to derive the final results for microtasks. We considered three baselines: 1) RandomMV: This approach used a random strategy for task assignment and aggregated workers' answers to generate the result by using majority voting. 2) RandomEM: The approach also used the random strategy for task assignment, but an Expectation-Maximization (EM) algorithm [31, 8] was used to aggregate the answers. The EM algorithm iteratively estimated worker's accuracy (called *confusion matrix* in [31, 8]) and exploited the estimated accuracy to compute the aggregated result. 3) AvgAccPV: This approach estimated an average accuracy for each worker by injecting some "gold" microtasks with ground truth, assigned microtasks to workers with higher accuracies, and aggregated the answers using the probabilistic verification model introduced in [22].

Evaluation metrics: We evaluated approaches on accuracy and efficiency. Accuracy was computed as the ratio of the microtasks with correctly predicted results, and the efficiency was measured by the elapsed time of task assignment.

Experiment settings: All approaches were evaluated under assignment size for each microtask $k = 3$. In addition, all the programs were implemented in JAVA and all the experiments were run on a Ubuntu machine with an Intel Core 2 Quad X5450 3.00GHz processor and 4 GB memory.

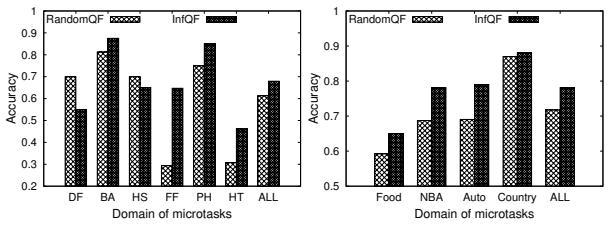
6.2 Diverse Accuracies across Domains

We first investigated the diversity of workers' accuracies across different domains by analyzing the collected answers in each dataset. For each worker, we computed the number of her completed microtasks and the accuracy (the ratio of correct answers to all of her completed microtasks) in each domain. Figure 6 shows the results on the two datasets, where the length of a bar represents the accuracy (the actual value of the accuracy is also labelled beside the bar).

We had the following observations of diversity. On the one hand, accuracies of each individual worker might be rather diverse across domains: workers could be very good in some domains, while performing badly in others. For example, as shown in Figure 6(a), the first worker A2YEBGPVQ41ESM had good accuracies in domains Books&Authors (0.875) and Philosophy (0.70) but had rather bad accuracies in Diet&Fitness (0.35), Home Schooling (0.30), Hunting (0.231), and FIFA (0.176). On the other hand, top workers w.r.t accuracy in different domains might also be diverse. As shown in Figure 6(b), the best worker in the Country domain was A2V99E4YEP14RI with accuracy 0.95. However, she was low-ranked in the NBA domain (due to the low accuracy 0.52), while the best worker in the NBA domain was worker A3JOGMTOAUEFUP. These observations have verified our claim in Section 1 that workers may have diverse accuracies across different tasks, and an average accuracy would be limited in reflecting worker's performance. These results also implied the necessity of our microtask assignment that took accuracy diversity into consideration, because some good workers (e.g., worker A2V99E4YEP14RI) in one domain (e.g., Country) could be a spammer in another domain (e.g., NBA).

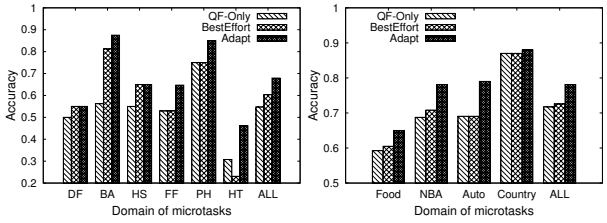
6.3 Evaluation on Adaptive Crowdsourcing

Next, we evaluated our adaptive crowdsourcing approach on accuracy in this section. We first examined the effect of the parameters used in our approach, i.e., microtask similarity measures, parameter α in Equation (2) and assignment size k, and then evaluated the approximation error of our greedy algorithm for adaptive assignment. Due to the space



(a) YahooQA dataset. (b) ItemCompare dataset.

Figure 7: Effect of Qualification.



(a) YahooQA dataset. (b) ItemCompare dataset.

Figure 8: Effect of The Adaptive Strategy.

limitations, we put these results in Appendix D. Here we evaluated the effect of qualification and adaptive assignment.

6.3.1 *Effect of Qualification*

We evaluated the effect of qualification and compared two qualification strategies. The first one, denoted by `RamdomQF`, randomly selected a set of `Q` qualification microtasks, while the second one, denoted by `InfQF`, utilized the techniques introduced in Section 5 for qualification selection. In the experiments, we set the number `Q` as 10.

Figure 7 provides the experimental results. We can see that `InfQF` outperformed `RamdomQF` in most of the domains and the overall cases. For example, as shown in Figure 7(a), the accuracy of `InfQF` was higher than that of `RamdomQF` in four out of six domains (`Books&Authors`, `FIFA`, `Philosophy` and `Hunting`), and 8% higher in the overall case (`ALL`). In addition, `InfQF` outperformed `RamdomQF` in all the domains and the overall case on the `ItemCompare` dataset (Figure 7(b)). The main reason is that the qualification microtasks selected by `RamdomQF` might be scattered in different domains, and the number of qualification microtasks in each individual domain would thus be limited. This resulted in overestimation or underestimation of workers' accuracies in the individual domains. In contrast, the `InfQF` method was more "focused", as it would select more qualification microtasks that had large influence to individual domains. These more focused qualification microtasks were helpful to reduce estimation errors and eliminate "bad" workers in the early qualification phase. For instance, on the `YahooQA` dataset, `InfQF` identified and eliminated workers with limited accuracies in the `FIFA` domain, e.g., workers `A2YEBGPVQ41ESM` and `A1H8Y5D04A7T5E` shown in Figure 6(a), and achieved more than 30% improvement on accuracy compared with `RamdomQF` in the domain. In addition, we also observed that `RamdomQF` might select good workers via qualification in some special cases (e.g., `Diet&Fitness` and `Home Schooling` in Figure 6(a)), as it might select some qualification microtasks, which were good for these domains, by chance. However, the method had unsatisfactory performance in most of the cases and achieved low overall performance.

In the remaining part of this section, we use `InfQF` as the default method for qualification selection.

6.3.2 *Effect of Adaptive Assignment*

Then, we evaluated the effect of our adaptive microtask assignment strategy. We studied the following two alternative strategies: 1) `QF-Only`: this strategy utilizes the accuracy which is estimated by qualification microtasks for assignment and does not adaptively update accuracy estimation. 2) `BestEffort`: the strategy, although adaptively updates accuracy estimation, adopts a best effort method for microtask assignment: given a request from a worker, it compares the estimated accuracies of the microtasks the worker could work on and assigns the one with the highest accuracy. 3) `Adapt`: this strategy not only adaptively updates accuracy estimation using our graph-based estimation techniques introduced in Section 3, but also assigns microtasks based on our adaptive method proposed in Section 4.

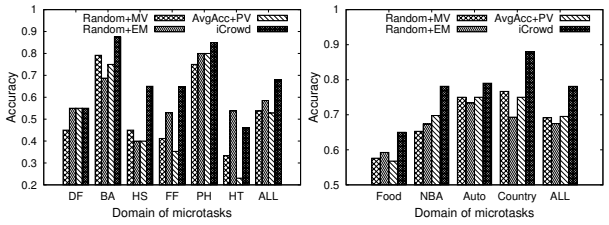
Figure 8 shows the experimental results. We can see that `QF-Only` achieved the worst performance in most of the cases. This is due to the estimation error in qualification: as qualification microtasks only took a small portion of the dataset, estimation from these microtasks might not be very accurate. `BestEffort` improved the accuracy compared with `QF-Only` in many domains, because it adaptively updated accuracy estimation as workers completed microtasks. However, the improvement was not significant. This is mainly due to its microtask assignment strategy. The best microtask (with the highest accuracy) of a worker might not take the worker as the best candidate to complete the task, because there might be other better workers. As such, this assignment strategy introduced workers with relatively low accuracy and thus affected the majority voting result, which would also have negative influence on the following accuracy estimation. `Adapt` achieved the best accuracy on both datasets. The improvement was attributed to not only our adaptive accuracy estimation but also the assignment strategy that took optimal microtask assignment and worker accuracy testing. In the remainder of this section, we utilize `Adapt` as the default strategy of *i*`Crowd`.

6.4 Comparison with Existing Approaches

We compared *i*`Crowd` with the three baseline approaches mentioned in Section 6.1, `RandomMV`, `RandomEM` and `AvgAccPV`. We used the same set of microtasks for qualification.

Figure 9 shows the performance of these approaches on accuracy. We can see that the performance of random assignment strategies `RandomMV` and `RandomEM` largely depended on the quality of the worker set in a domain: if a domain contained many more good workers, the accuracy would be high; otherwise, the accuracy would be low. For example, as shown in Figure 6(a), the average worker quality of the `FIFA` domain was much lower than that of the `Philosophy` domain, and thus the random strategies achieved much worse result in `FIFA` (only about 0.4 on accuracy). We also had an interesting observation that the EM algorithm for worker answer aggregation might not always be helpful. For instance, the accuracy of `RandomEM` was lower than that of `RandomMV` in the domains `Auto` and `Country` in Figure 9(b). This was mainly attributed to the accuracy estimation mechanism of EM. The algorithm estimated worker accuracies without considering the inherent accuracy diversity over domains, and thus might overestimate some workers with limited accuracies. The iterative mechanism of EM might further propagate and aggravate the overestimation.

Moreover, `AvgAccPV` achieved limited performance on the two datasets, which verified our claim that the average accu-



(a) `YahooQA` dataset. (b) `ItemCompare` dataset.

Figure 9: Approach Comparison.

racy without considering domain diversity may not be effective for microtask assignment. For example, `AvgAccPV` would incorrectly estimate that the first worker `A2YEBGPVQ41ESM` in Figure 6(a) with high average accuracy also had good performance in the `Hunting` domain, while the worker was bad in `Hunting`. This resulted in unsatisfactory accuracy of `AvgAccPV` in this domain (only 25% as shown in Figure 9(a)).

Our approach *i*`Crowd` outperformed the baseline approaches. As shown in Figure 9, it gained about 10% improvement in the average case (the `ALL` domain), and even more than 20% improvement in some domains (e.g., `Home Schooling`). The main reason was that *i*`Crowd` utilized the graph-based estimation model for capturing accuracy diversity in different domains, which was very helpful to identify the workers with expertise in each individual domain. Furthermore, the assignment strategy of *i*`Crowd` could effectively assign microtasks to high-quality workers and progressively improved accuracy estimation based on worker answers. We also observed that *i*`Crowd` had limited improvement in the `Auto` domain on the `ItemCompare` dataset. This is mainly because there was no very good workers in this domain: as shown in Figure 6(b), the best worker in `Auto` only had an accuracy of 0.76, while accuracies of the best workers in other three domain were more than 0.9. As a result, *i*`Crowd` might not have much space for choosing high-quality assignments.

6.5 Evaluation on Efficiency

We evaluated efficiency of our assignment algorithm using simulation. Initially, the entire microtask set was empty. We inserted 0.2 million microtasks at each time and ran *i*`Crowd` to evaluate the efficiency. We also considered the maximal number of "neighbors", which can be influenced by a microtask in our accuracy inference. Given a maximal neighbor number, say 40, and a microtask, we randomly selected 40 microtasks as neighbors of the microtask. Figure 10 shows the experimental results. We can see that our method was efficient and scaled very well: with the increase of the numbers of microtasks, the elapsed time increased sub-linearly. The good performance on efficiency was mainly because we devised effective index structures and developed efficient algorithms for microtask assignment.

7. RELATED WORK

The studies most related to our work are quality-control techniques in crowdsourcing. The existing approaches can be broadly classified into two categories. The *gold-injected* approaches [22, 1] leveraged a small amount of tasks with ground truth to estimate workers' reliability and aggregated their answers based on the reliability. The *EM*-based approaches [31, 8, 18, 21] simultaneously estimated worker accuracy and predicted aggregated results using an Expectation-Maximization (EM) strategy. Moreover, Ipeirotis et al. [17] studied how to evaluate the quality of workers to reject or

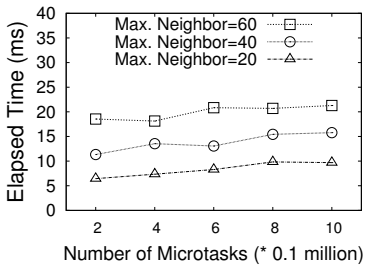


Figure 10: Evaluating Scalability with Simulation.

block low-quality workers. Parameswaran et al. [26] assumed the false positive and false negative rates of each worker, and studied the number of assignments that are required for a task. Our study is different from these quality-control techniques in that we tackle the accuracy diversity in more complicated crowdsourcing tasks for better accuracy estimation and *adaptively* assign tasks to workers, instead of random task assignment without considering workers' expertise.

Recently, some approaches for task assignment [15, 16, 19, 20] and recommendation [3] have been proposed. Although they also considered how to recommend and assign tasks to appropriate workers, they neglected the fact that different works had accuracy diversity across different tasks. Thus different from these studies, we attempted to address two new challenges. First, we studied how to effectively estimate the diverse accuracy of workers based on their completed tasks. Second, we proposed efficient algorithms to support instant task assignment based on workers' divserse accuracy.

There are some crowd-powered systems developed in the human-computer interaction community. Solyent [5] is a word processor that employs the *Find-Fix-Verify* interaction method. Adrenaline [4] is a crowd-powered camera that supports realtime crowdsourcing by using the interaction method called *rapid refinement*. Recent studies in the database community aim to leverage crowdsourcing to build database systems, which can support some queries that cannot be processed by machine-only methods (e.g. entity resolution and graph search) [25, 29, 12, 23, 27, 28, 32, 9, 34, 13, 10]. Nevertheless, *i*`Crowd` has a different goal from these studies: *i*`Crowd` focuses on adaptively assigning tasks to appropriate workers to improve the crowdsourcing quality.

8. CONCLUSION

We proposed *i*`Crowd`, an adaptive crowdsourcing framework to adaptively assign microtasks to appropriate workers who have high probabilities to correctly complete microtasks. We devised a graph-based estimation model to capture the diverse accuracies of workers across different tasks in order to provide more effective task assignment. We proved that the optimal task assignment is NP-hard and devised a greedy algorithm to enable instant task assignments. We deployed *i*`Crowd` on AMT and experimental results on two real datasets show that *i*`Crowd` achieved much higher quality than state-of-the-art methods.

Acknowledgement: This work was partly supported by the 973 Program of China (2015CB358700), a Singapore Ministry of Education AcRF Tier 1 Grant No. R-252-000-513-112, the "NExT Research Center" (WBS: R-252-300-001-490), the NSFC project (61422205, 61472198), the Chinese Special Project of Science and Technology (2013zx01039-002-002), Tencent, Huawei, SAP, YETP0105, and the FDCT/106/2012/A3.

9. REFERENCES

[1] http://crowdflower.com/.

[2] Amazon Mechanical Turk. https://www.mturk.com.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[4] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger. Crowds in two seconds: enabling realtime crowd-powered interfaces. In *UIST*, pages 33–42, 2011.

[5] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *UIST*, pages 313–322, 2010.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. In *NIPS*, pages 601–608, 2001.

[7] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. *PVLDB*, 5(11):1495–1506, 2012.

[8] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 28(1):20–28, 1979.

[9] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.

[10] J. Fan, M. Lu, B. C. Ooi, W. Tan, and M. Zhang. A hybrid machine-crowdsourcing system for matching web tables. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 976–987, 2014.

[11] A. Feng, M. J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, and R. Xin. Crowddb: Query processing with the vldb crowd. *PVLDB*, 4(12):1387–1390, 2011.

[12] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, pages 61–72, 2011.

[13] S. Guo, A. G. Parameswaran, and H. Garcia-Molina. So who won?: dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396, 2012.

[14] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526, 2002.

[15] C. Ho, S. Jabbari, and J. W. Vaughan. Adaptive task assignment for crowdsourced classification. In *ICML*, pages 534–542, 2013.

[16] C. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.

[17] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *SIGKDD Workshop on Human Computation*, pages 64–67, 2010.

[18] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, pages 1953–1961, 2011.

[19] D. R. Karger, S. Oh, and D. Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

[20] H. W. Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. 2010.

[21] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In *NIPS*, pages 701–709, 2012.

[22] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. Cdas: A crowdsourcing data analytics system. *PVLDB*, 5(10):1040–1051, 2012.

[23] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[24] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, pages 211–214, 2011.

[25] A. G. Parameswaran, S. Boyd, H. Garcia-Molina, A. Gupta, N. Polyzotis, and J. Widom. Optimal crowd-powered rating and filtering algorithms. *PVLDB*, 7(9):685–696, 2014.

[26] A. G. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, pages 361–372, 2012.

[27] A. G. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR*, pages 160–166, 2011.

[28] A. G. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5):267–278, 2011.

[29] H. Park and J. Widom. Crowdfill: collecting structured data from the crowd. In *SIGMOD*, pages 577–588, 2014.

[30] V. C. Raykar and S. Yu. Ranking annotators for crowdsourced labeling tasks. In *NIPS*, pages 1809–1817, 2011.

[31] V. S. Sheng, F. J. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *SIGKDD*, pages 614–622, 2008.

[32] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsoucing entity resolution. *PVLDB*, 5(10), 2012.

[33] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: How similar is similar. *PVLDB*, 4(10):622–633, 2011.

[34] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. Technical report, Stanford University.

[35] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

APPENDIX

A. IMPLEMENTATION OF ICROWD ON AMAZON MECHANICAL TURK

We developed a system to support adaptive crowdsourcing on Amazon Mechanical Turk (AMT), as shown in Figure 11. As AMT does not support advanced HIT assignment (it only randomly assigns HITs to workers), the system utilized the *ExternalQuestion* mechanism[3] to manage microtasks on our own Web server and take full control of microtask as-

[3]More information can be found in AMT Documentation at http://aws.amazon.com/documentation/mturk/

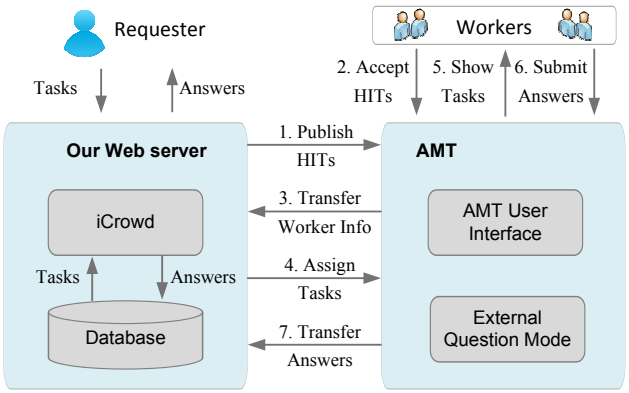


Figure 11: Implementation of *i*Crowd on AMT

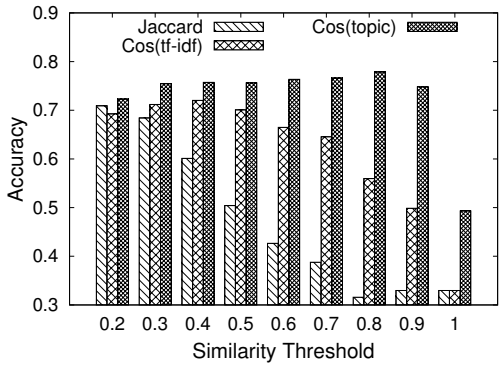


Figure 12: Evaluating Different Similarity Measures and Thresholds on The ItemCompare Dataset.

signments. Specifically, the system generates HITs with the ExternalQuestion mode, which are different from the ordinary HITs in that each of such HITs only contains a URL of our Web server, instead of the actual microtasks. Then, when a worker accepts a published HIT and requests for microtasks[4], AMT will send a request containing worker information to our server. Given the worker information, *i*Crowd in our server assigns microtasks to the worker and sends them back to AMT. Next, AMT embeds the assigned microtasks as an *HTML iframe* in its user interface to present the microtasks to the worker. After a worker submits her answers, AMT transfers the answers to our server and our server calls back some APIs of AMT to process payment. Finally, after collecting answers for all the microtasks, *i*Crowd aggregates the answers and stores the obtained result. Briefly speaking, when a worker requests a microtask, our system decides how to assign microtasks to the worker; when a worker submits an answer, it collects the answer, aggregates the answers, and computes which domains each worker is skilled in. Iteratively, our system can assign all tasks to appropriate workers.

B. PROOF OF LEMMA 4

We can prove the lemma by a reduction from the k-set packing problem, which is known to be NP-hard.

Recall that an instance of k-set packing problem $(\mathcal{U}, \mathcal{C}, k)$ consists of a universe of elements $\mathcal{U} = \{e_1, e_2, \ldots, e_{|\mathcal{U}|}\}$, a collection of subsets $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$ where $C_i \subseteq \mathcal{U}$, and a number k. Each subset C_i contains less than k elements, i.e., $|C_i| \leq k$, and it is associated with a weight $w(C_i)$. The problem aims to select some subsets $\mathcal{C}^* \subseteq \mathcal{C}$ satisfying $\forall C_i, C_j \in \mathcal{C}^*$ are disjoint, to maximize $\sum_{C_i \in \mathcal{C}^*} w(C_i)$.

Given any instance $(\mathcal{U}, \mathcal{C}, k)$ of the k-set packing problem, we create an instance of the microtask selection problem as follows. We first create a set of active workers $\mathcal{W}$, each of which corresponds to an element in $\mathcal{U}$. Then, we create a set of microtasks $\mathcal{T}$, each of which corresponds to a subset in $\mathcal{C}$, and let the number of assignments for each microtask be equal to k. Then, it is easy to prove that the optimization objectives of the problems are identical. The problem of microtask selection can be solved, only if the k-set packing problem is solved. Thus, we prove the lemma.

[4]Usually a HIT contains mutiple microtasks. For each HIT, we only need to tell the worker how many microtasks in the HIT and show the first microtask in the HIT. When the worker finishes the microtask and clicks the "Next" link, we assigns the next microtask to the worker.

C. PROOF OF LEMMA 5

We can prove the lemma by a reduction from the maximum coverage problem, which is known to be NP-hard.

Recall that an instance of maximum coverage problem $(\mathcal{U}, \mathcal{C}, k)$ consists of a set of elements $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$, a collection of subsets $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$ where $C_i \subseteq \mathcal{U}$, and a number k. The problem aims to select k subsets $\mathcal{C}^* \subseteq \mathcal{C}$ to maximize the number of covered elements, $|\bigcup_{C \in \mathcal{C}^*} C|$.

Given any instance $(\mathcal{U}, \mathcal{C}, k)$ of the maximum coverage problem, we create an instance of the qualification microtask selection problem as follows. We first create a set of microtask $\mathcal{T}$, each of which corresponds to an element in $\mathcal{U}$. Then, we create accuracy vector $\mathbf{p}_{t_i}$ corresponding to each set $C \in \mathcal{C}$: for any $u_i \in C$, we set the $p_i = 1$; otherwise $p_i = 0$. Next, we let the number of qualification microtasks be equal to k. Then, it is easy to prove that the problem of qualification microtask selection can be solved, only if the maximum coverage problem is solved. Thus, we prove the lemma.

D. ADDITIONAL EXPERIMENTS

D.1 Evaluation on Similarity Measures

We investigated the methods for measuring microtask similarity, which were essential to generate the similarity graph for accuracy estimation. To compute microtask similarity, we tokenized the text of microtasks and removed the stopwords. Then, given two microtasks, we considered the following three similarity measures: 1) `Jaccard`: this method took each microtask as a set of words and computed similarity of two microtasks as the their overlap size divided by their union size; 2) `Cos(tf-idf)`: this method took each microtask as a vector of words associated with TF-IDF weights, then it computed the similarity as the cosine similarity of the vectors; 3) `Cos(topic)`: this method first conducted *topic analysis* by using Latent Dirichlet Allocation (LDA) [6] to obtain a topic distribution for each microtask, and then computed microtask similarity via the cosine similarity of the two distributions. We also set a similarity threshold and only utilized the microtask pairs with similarity larger than the threshold.

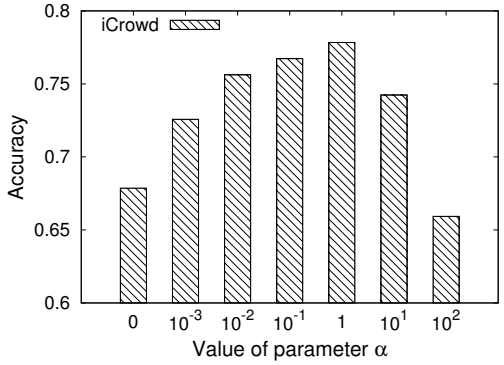


Figure 13: Evaluating Parameter α on The `ItemCompare` Dataset.

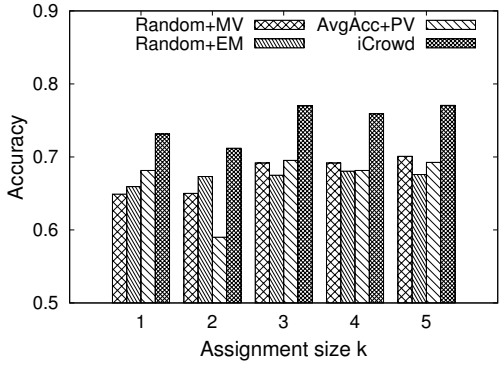


Figure 14: Evaluating Assignment Size k on The `ItemCompare` Dataset.

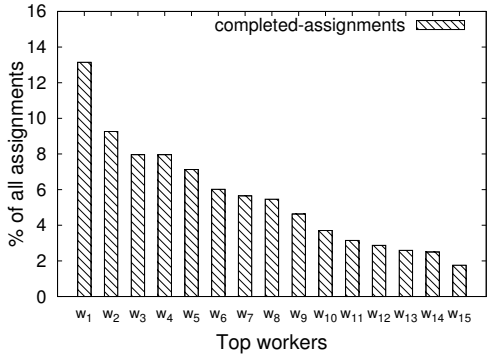


Figure 15: Distribution of Microtasks Completions for Top Workers on The `ItemCompare` Dataset.

The experimental result is shown in Figure 12. We have the following observations. First, different similarity measures did not significantly affect the performance. For example, for a small similarity threshold, the three methods achieved similar performance. Second, the similarity thresholds had effect on the performance. A small threshold involved some weak microtask connections and a large threshold eliminated many strong microtask connections. Thus an appropriate threshold could be helpful to achieve high performance. Third, `Cos(topic)` achieved better performance, because the topic analysis from the LDA model could discover the inherent topical relevance between microtasks in the same domain while differentiate microtasks in different domains.

We used `Cos(topic)` with threshold 0.8 as the default similarity measure and threshold for all of our experiments.

D.2 Evaluation on Parameter α

We evaluated the effect of α introduced in Equation (2). This parameter was used to leverage the two objectives in accuracy estimation: 1) minimizing accuracy difference between similar microtasks, and 2) minimizing the difference between estimated accuracies and observed accuracies. We varied α and examined the accuracy of our approach.

As shown in Figure 13, depending solely on one objective could not achieve satisfactory accuracy: On the one hand, only depending on the first objective ($\alpha = 0$) would lead to all connected microtasks having the same estimated accuracies, which failed to capture accuracy diversity. On the other hand, only depending on the second objective (α is a large number, say 100) would make estimated accuracies very close to the observed ones, which means the estimation could not benefit from the graph-based inference. In contrast, reconciling these two objectives (i.e., finding an appropriate α) produced much better accuracy.

W set $\alpha = 1.0$ as the default value for all of our experiments.

D.3 Evaluation on Assignment Size k

We evaluated the approaches, `RandomMV`, `RandomEM`, `AvgAccPV`, and *i*`Crowd`, by varying the assignment size k. As mentioned in Section 2.1, this parameter is used to set the number of workers which are assigned with each microtask.

Figure 14 shows the experimental result. We can see that our proposed approach *i*`Crowd` achieved the highest accuracy at every assignment size k, and outperformed the three baselines. This is because *i*`Crowd` could effectively estimate workers' diverse accuracies and, given each k, assign top-k workers with higher accuracies to each microtask. Besides, we observe that increasing k would generally improve the overall accuracy. For example, accuracy of *i*`Crowd` was improved by 5 percent from $k = 1$ to $k = 3$. This illustrates that voting among more workers could increase the accuracy. Nevertheless, the improvement might become insignificant given a larger assignment size. This is because the newly assigned workers have relatively lower estimated accuracies and thus have lower weights in the voting to affect the final result.

D.4 Approximation Error of Greedy Algorithm for Microtask Assignment

We evaluated our greedy algorithm (Algorithm 3) and examined its approximation error on the larger dataset `ItemCompare`. Specifically, we implemented an enumeration-based algorithm to find the optimal solution with the maximum overall accuracy, denoted by OPT: it enumerated all feasible assignment schemes and returned the one with the maximum overall accuracy. Meanwhile, we ran our greedy algorithm to produce a solution with overall accuracy denoted by APP. Then, we measured the approximation error using the equation

$$Error = \frac{OPT - APP}{OPT} \times 100\%. \quad (6)$$

Table 5 shows the result where we varied the number of active workers to be assigned (i.e., $\mathcal{W}$ in Algorithm 3). Notice that the enumeration-based algorithm took very long

Table 5: Approximation Errors of Algorithm 2 on The `ItemCompare` Dataset.

# active workers	3	4	5	6	7
Approx. Error (%)	1.9	0.6	0.5	1.3	1.5

computation time when the number of active workers was larger than 7 (we waited 30 minutes and did not get the result). This is because, in this case, there were 337 microtasks having non-empty top worker sets ($\{\widehat{\mathcal{W}}(t_i)\}$). The optimal algorithm had to enumerate all possible subsets of 337 microtasks such that the top worker sets in each of such subsets were disjoint, which was rather time-consuming. As such, we only reported the results for the number of active workers up to 7. As shown in Table 5, the approximation errors of our greedy algorithm were less than 2%.

This illustrated that the performance of the greedy algorithm was very close to that of the optimal task assignment.

D.5 Assignment Distribution w.r.t Workers

In this section, we examined the distribution of the number of microtask assignments to workers on the larger `ItemCompare` dataset. This dataset had 360 microtasks and the assignment size for each microtask was $k = 3$. Thus the number of total assignments was 1080.

Figure 15 shows the result for the top-15 workers. We can see that each of these worker completed a fair number of assignments, e.g., the first worker w_1 completed more than 13% of all the assignments, and these top-15 workers completed 84% of all the assignments. This observation supports our claim that the worker set that completed a crowdsourcing job was relatively stable.