Comlet: A Crowdsourcing Multi-Label Task System

Yudian Zheng[†], Guoliang Li[*], Reynold Cheng[†], Huiqi Hu[‡]
[†]The University of Hong Kong, [*]Tsinghua University, [‡]East China Normal University
ydzheng2@cs.hku.hk, liguoliang@tsinghua.edu.cn, ckcheng@cs.hku.hk, hqhu@sei.ecnu.edu.cn

Abstract*—Crowdsourcing platforms, such as the Amazon Mechanical Turk (AMT), can harness the power of human effort to solve problems that are difficult for computers (e.g., sentiment analysis and image labeling). A fundamental crowdsourcing task is *labeling*, where a crowd worker is asked to answer the task, by selecting the correct label(s) among multiple given choices. For this kind of tasks, previous works often assume that a worker can only select a single label. However, this is inadequate in many important applications (e.g., image tagging), where multiple labels (e.g., items that appear in an image) are acquired from the crowd. In this paper, we study the *multi-label* task, where a worker can select more than one label. In particular, we investigate the *truth inference problem*: Given workers' answers to a multi-label task, what are the correct labels of this task? We address this issue by modeling workers' behaviors in performing multi-label tasks. We design an iterative algorithm to effectively infer each task's correct labels. We then study how to speed up such computation. We also explore the use of correlations among different labels to improve the inference quality. We devise an online task assignment algorithm, which decides how to instantly assign tasks to appropriate workers. Based on these solutions, we built Comlet, a Crowdsourcing Multi-Label Task System. We have performed extensive evaluations of Comlet on two real crowdsourcing platforms, i.e., AMT and ChinaCrowd. Our results show that Comlet outperforms state-of-the-art approaches, and is robust under different scenarios.

I. Introduction

Crowdsourcing solutions have been proposed to solve problems that are hard for computers. Consider a sentiment analysis problem [34,55], where a product company collects many reviews from users on its products, and it aims to know users' sentiments about the reviews. Existing algorithms often cannot compute sentiments accurately [33]. Crowdsourcing solutions can be used, where for each review, we generate a task and ask the crowd to label the sentiment (e.g., selecting a label from "*positive*", "*neutral*" or "*negative*").

Existing crowdsourcing studies [29,34,48,53,55] focus mainly on single-label tasks, which require workers to select a single label (or choice), e.g., select one label from $\{$*positive*, *neutral*, *negative*$\}$ in a sentiment analysis task. However, an object can have multiple labels. For example, an image in Figure 1 has *tree*, *sky*, and *mountain*; a movie "Matrix" can be labeled with *action* and *sci-fi*. Although we can transform a multi-label task to several single-label tasks, this simple approach can generate many tasks, incurring a high cost and latency. For example, the multi-label task in Figure 1 is transformed to 10 single-label tasks, where each task inquiries about whether or not the image contains a certain label (e.g., *tree*). As discussed in [21], compared with single-label tasks, multi-label tasks enable six times of improvement in terms of human computation time, without sacrificing the quality.

Although some recent works [15,23,38,39,47] focus on solving multi-label tasks in crowdsourcing, this problem is not well addressed. Workers may have different characteristics in multi-label tasks: a conservative worker would only select labels that the worker is certain of, while a venturous worker may select more labels. In order to determine the multi-label tasks' results, the key is to devise the so-called "*worker model*" to accurately express the behavior of the worker in answering multi-labels. Then we can identify the most trustworthy workers based on their models and put trust in their answers. However, existing works [15,23,38,39,47] simply adapt the worker models in modeling workers' behaviors for single-label tasks [19,29,34,39,53,55], which are not suitable in our scenario. In a multi-label task, there can be multiple labels for the worker. Consider the case where there are 20 labels, and only four of them are correct (i.e., the remaining 16 labels are wrong). A "bad" worker may select none of the correct labels, and instead submit a few (say two) wrong labels. Under the *true negative rate* (TNR) [29,39,55], his/her quality score is computed as $(16-2)/16 = 87.5\%$. But the extremely high score of this worker cannot reflect his/her poor performance! The consequence is that we may be misled to believe in a bad answer. As we will explain, *none* of the existing worker models are suitable for multi-label tasks.

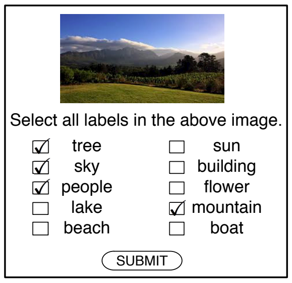
Furthermore, different from single-label tasks, the inherent correlations among labels exist in multi-label tasks. For example, in the 10 labels of Figure 1, consider a pairwise label dependency: if an image has label *sun*, then it is highly probable that it also has label *sky*. Similar correlations also hold for labels *boat* and *lake*. Label correlations can be regarded as *prior* information, which can be learned from existing machine learning works [52,54]. The label correlations can be used to improve the inference quality.

To address these limitations, in this paper we perform a comprehensive investigation for the multi-label tasks. To be specific, we have developed a Crowdsourcing Multi-Label Task System, called Comlet. Its framework is shown in Figure 2, where task publisher(s) can deploy multi-label tasks on our system. Comlet interacts with crowdsourcing platforms, e.g., Amazon Mechanical Turk (AMT) [1], from which it collects workers' answers, stores them in database and conducts inference. It can also wisely select tasks to assign upon a worker's request. In summary, there are two kinds of requests from workers that need to be handled by Comlet: (1) when a worker submits answers to Comlet; (2) when a worker comes and requests tasks. We develop two components in Comlet that process these two requests, respectively:

- **Truth Inference.** When a worker submit answers, the

TABLE I
An Example of Objects and Candidate Label Sets.

object	o_1=	o_2=
candidate label set	L_1={tree, sky, people, lake, beach, sun, building, flower, mountain, boat}	L_2={tree, sky, people, lake, beach, sun, building, flower, mountain, boat}
correct labels	{tree, sky, mountain}	{sky, lake, sun, boat}



Select all labels in the above image.

☑ tree	☐ sun
☑ sky	☐ building
☑ people	☐ flower
☐ lake	☑ mountain
☐ beach	☐ boat

SUBMIT

Fig. 1. An Example Task.

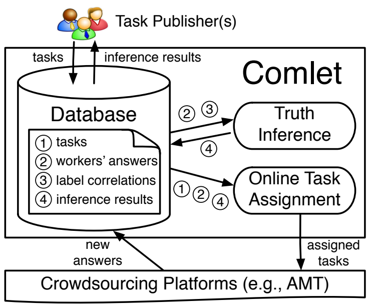


Fig. 2. The Comlet Framework.

component infers the truth (or correct labels) of each task based on all workers' answers. Comlet uses a novel worker model, which can capture workers' diverse characteristics in answering multi-label tasks. As the truth of each task is unknown, each worker's model can only be estimated based on the collected answers. Comlet conducts truth inference in an iterative approach, which can jointly infer all tasks' truth and workers' models with the following principle: a worker that selects correct labels often will be considered to have a higher quality; meanwhile, a label that is selected by high quality workers for a task is likely to be a correct label for the task. We also study how to speed-up the computation by designing an incremental approach. Moreover, Comlet leverages the known label correlations to improve the truth inference, by integrating them into the inference method.

• **Online Task Assignment.** When a worker requests tasks, the component targets at instantly assigning k tasks to the worker. A poor assignment may not only waste the budget and time, but also spoil the overall quality. Comlet first measures the ambiguity of each task based on the collected answers, and then estimates how much ambiguity can be reduced *if* the task is really answered by the worker. Finally the k tasks with the highest reduction in ambiguity will be assigned. As the worker's answer to the task is unknown, to compute its reduction in ambiguity, all possible answers given by the worker should be considered, which is exponential (e.g., 2^ℓ answers for ℓ labels). Moreover, to select k tasks (say, out of n tasks), we have to consider all $\binom{n}{k}$ combinations. To reduce the computational complexity, we prove a theorem, which computes the optimal assignment in linear time.

To summarize, our contributions are: (1) we perform a comprehensive study on crowdsourcing multi-label tasks, by studying the *Truth Inference Problem* and *(Online) Task Assignment Problem* (Section II). (2) For the first problem, we propose an effective worker model (Section III), and devise both iterative and incremental methods that jointly infer each task's truth and each worker's model (Section IV). We further consider how to integrate label correlations into our method (Section V). (3) For the second problem, we develop an effective algorithm that judiciously selects k tasks with the largest amount of ambiguity reduction for the current worker, in linear time (Section VI). (4) We have developed Comlet, and use two real-world datasets to perform experiments on two crowdsourcing platforms (AMT [1] and ChinaCrowd [2]). Results show that Comlet outperforms existing state-of-the-art methods, and is robust under various scenarios. It achieves over 20% improvements on the two datasets performed by low-quality workers. We also conduct experiments on simulated data, in order to verify the scalability of Comlet (Section VII).

II. The Multi-Label Problem

A. Data Model

Definition 1 (Object, Candidate Label Set): Given n objects, $o_1, o_2, \cdots, o_n$, where each object o_i has a candidate label set L_i=$\{\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,|L_i|}\}$, our target is to select the correct labels of each object o_i from its candidate label set L_i.

For example, Table I shows two objects o_1, o_2 and their corresponding candidate label sets L_1 and L_2. Note that there are several options to generate candidate labels: (1) they can be collected from user data, e.g., user image tags in Flickr [6]; (2) we can leverage existing computer vision system (e.g., ConvNet [3]), where given an image, it can estimate the probability of each label being the correct label among a large set of labels, and we can keep the labels with high probabilities as our candidate labels. The correct labels for o_1 and o_2 are $\{\texttt{tree}, \texttt{sky}, \texttt{mountain}\}$ and $\{\texttt{sky}, \texttt{lake}, \texttt{sun}, \texttt{boat}\}$, respectively. To effectively obtain the correct labels of each object, we resort to crowdsourcing, by asking workers to select labels of each object from its candidate label set.

Definition 2 (Task, Worker, Answer): A task contains an object o_i and its candidate label set L_i. It asks the workers to select correct labels for o_i from L_i. Let $\mathcal{T} = \{(o_1, L_1), (o_2, L_2), \ldots, (o_n, L_n)\}$ denote the set of tasks. To tolerate errors from the workers, each task can be answered by multiple workers. We denote $\mathcal{W}$ as the set of all workers, and W_i as a set of workers that answer object o_i.

Example 1: Based on Table I, we can generate two tasks $\mathcal{T} = \{(o_1, L_1), (o_2, L_2)\}$. We show the first task in Figure 1, which contains an image (o_1) and a set of 10 labels (L_1). Workers will identify which labels the image has, by ticking '✓' to the corresponding labels. Table II shows workers' answers, where each row represents a worker's selected labels (or answer) for an object. For example, the first row indicates that worker w_1 only selects $\{\texttt{sky}\}$ for o_1. We can see that o_1 is answered by all 4 workers, and o_2 is answered by 3 workers. Thus $\mathcal{W} = W_1 = \{w_1, w_2, w_3, w_4\}$, and $W_2 = \{w_1, w_2, w_3\}$.

Next, for each pair $(o_i, \ell_{i,j})$, we define its collected answers (called votes) from workers and its ground truth (called truth).

Definition 3 (Vote, Truth): For a pair $(o_i, \ell_{i,j})$ $(1 \le i \le n$, $1 \le j \le |L_i|)$, a vote $v^w_{i,j}$ = Y/N represents whether or not worker w selects label $\ell_{i,j}$ for o_i. To be specific, $v^w_{i,j}$ = Y (N) means that worker w selects (does not select) label $\ell_{i,j}$ for object o_i. For a pair $(o_i, \ell_{i,j})$ $(1 \le i \le n, 1 \le j \le |L_i|)$, we denote its truth as $t_{i,j}$ = Y/N, which represents whether or not label $\ell_{i,j}$ is a correct label for object o_i, i.e., $t_{i,j}$ = Y (N) means that label $\ell_{i,j}$ is (not) a correct label for object o_i.

Example 2: Consider the two tasks in Table I and workers' answers in Table II, we generate a new Table III, which lists

TABLE II
WORKERS' ANSWERS FOR OBJECTS.

object	worker	answer
o_1	w_1	{sky}
o_1	w_2	{tree, sky, people, mountain}
o_1	w_3	{tree, flower}
o_1	w_4	{tree, sky, flower, mountain}
o_2	w_1	{sky, boat}
o_2	w_2	{sky, beach, sun}
o_2	w_3	{lake, sun, building}

TABLE III
TRUTH AND VOTES FOR ALL (OBJECT, LABEL) PAIRS.

(object, label) pair	the pair's truth	all workers' votes
$(o_1, \ell_{1,1})$	$t_{1,1} = \text{Y}$	$v^{w_1}_{1,1} = \text{N},\ v^{w_2}_{1,1} = \text{Y},\ v^{w_3}_{1,1} = \text{Y},\ v^{w_4}_{1,1} = \text{Y}$
$(o_1, \ell_{1,2})$	$t_{1,2} = \text{Y}$	$v^{w_1}_{1,2} = \text{Y},\ v^{w_2}_{1,2} = \text{Y},\ v^{w_3}_{1,2} = \text{N},\ v^{w_4}_{1,2} = \text{Y}$
$\cdots$	$\cdots$	$\cdots$
$(o_1, \ell_{1,10})$	$t_{1,10} = \text{N}$	$v^{w_1}_{1,10} = \text{N},\ v^{w_2}_{1,10} = \text{N},\ v^{w_3}_{1,10} = \text{N},\ v^{w_4}_{1,10} = \text{N}$
$(o_2, \ell_{2,1})$	$t_{2,1} = \text{N}$	$v^{w_1}_{2,1} = \text{N},\ v^{w_2}_{2,1} = \text{N},\ v^{w_3}_{2,1} = \text{N}$
$\cdots$	$\cdots$	$\cdots$
$(o_2, \ell_{2,10})$	$t_{2,10} = \text{Y}$	$v^{w_1}_{2,10} = \text{Y},\ v^{w_2}_{2,10} = \text{N},\ v^{w_3}_{2,10} = \text{N}$

the truth and votes for all $(o_i,\ \ell_{i,j})$ pairs where $1 \le i \le n$ and $1 \le j \le |L_i|$. In each row, for a given pair $(o_i,\ \ell_{i,j})$, we get its truth $t_{i,j}$ from Table I and all its votes from Table II. To make it simple, we adopt the label sequence of L_1 (L_2) as Table I, i.e., $\ell_{1,1} = \texttt{tree}$, $\ell_{1,2} = \texttt{sky}$, $\ldots$, $\ell_{1,10} = \texttt{boat}$. For example, for the pair $(o_1, \ell_{1,2})$: its truth $t_{1,2} = \text{Y}$, which means that label sky ($\ell_{1,2}$) is a correct label for o_1; it gets 4 votes from workers, e.g., $v^{w_3}_{1,2} = \text{N}$, which means that worker w_3 does not select sky ($\ell_{1,2}$) for object o_1.

B. Problem Definition

We now define the problems. The first important problem is to infer the truth (or correct labels of each object) based on workers' answers, called *Truth Inference Problem.*

Definition 4 (Truth Inference Problem): Given all workers' votes, i.e., $v^w_{i,j}$ for $1 \le i \le n$, $1 \le j \le |L_i|$, $w \in W_i$, infer the truth of each $(o_i, \ell_{i,j})$ pair, i.e., $t_{i,j}$.

To address this problem, we first propose a worker model to quantify a worker's quality on a task (Section III), and then devise methods to infer the truth of each task based on the worker model (Section IV). We discuss how to utilize the label correlation to improve the inference quality (Section V).

The second problem is an online problem, i.e., when a worker comes, how to judiciously select appropriate tasks to assign for the coming worker? Note that AMT [1] uses a unique ID (a string of 14 characters) to identify each worker and provide the *external-questions* [5] functionality, which enables us to deploy our server on AMT [1]. Then when a worker w comes, AMT will pass w's unique ID to us, and our sever can dynamically assign the selected tasks to w.

Definition 5 (Task Assignment Problem): Given all workers' answers for objects collected so far, when a worker comes, which k objects should be assigned to the coming worker?

We propose an effective algorithm that judiciously selects k tasks for each worker in linear time (Section VI).

III. WORKER MODELING

In this section, we first revisit different worker models (Section III-A), and then point out the limitations of models used in existing works and propose our selection (Section III-B).

To evaluate a worker's model (or quality), we should know the truth for all (object, label) pairs that the worker votes. For illustration purpose, in this section, we first assume that all truth have been known, and compare different ways of modeling a worker. Then, in the next section (Section IV), we show how to compute each worker's model without knowing the truth, i.e., in the real-world setting.

A. Worker Models Revisited

For each worker, based on all (object, label) pairs that the worker votes, and their truth, we can generate a contingency table for the worker in Table IV. In the table, v represents the given votes and t represents their truth, and both can be obtained from Table III. Then for each worker, based on Table III, we can compute four statistics: True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN). For example, TP of a worker is the number of (object, label) pairs that the worker votes Y and the truth is Y. In Table III, As worker w_3 has answered 2 objects, by answering only 5 pairs with Y, thus TP+FP=5. Among the five Y votes ($\texttt{tree}, \texttt{flower}$ for o_1, and $\texttt{lake}, \texttt{sun}, \texttt{building}$ for o_2), three of them are right, thus TP=3, FP=2. We also list all four workers' contingency-table values in Table V.

Based on the contingency table for a worker, there are several derivative parameters to model a worker's quality: Accuracy, Precision, Recall and True Negative Rate (TNR):
Accuracy. It is the probability that the worker's vote to an (object, label) pair is right: $\Pr(\ t = v\ ) = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}}$.
Precision. It is the probability that the worker's vote Y to an (object, label) pair is right: $\Pr(\ t = \text{Y} \mid v = \text{Y}\ ) = \frac{\text{TP}}{\text{TP+FP}}$.
Recall. It is the probability that an (object, label) pair with truth Y is rightly voted by the worker (i.e., the worker's vote is Y): $\Pr(\ v = \text{Y} \mid t = \text{Y}\ ) = \frac{\text{TP}}{\text{TP+FN}}$.
True Negative Rate (TNR). It is the probability that an (object, label) pair with truth N is rightly voted by the worker (i.e., the worker's vote is N): $\Pr(\ v = \text{N} \mid t = \text{N}\ ) = \frac{\text{TN}}{\text{TN+FP}}$.

Based on a worker's contingency table in Table V, the worker's different quality parameters can be derived in Table VI. For these parameters, existing works [29,34,39,53,55] either use only one or a combination of them to model a worker's quality: Accuracy is used in [34,53] to model a worker's quality, while [29,39,55] use a combination of TNR and Recall to model a worker's quality. In the following section (Section III-B), we first show the limitations of Accuracy and TNR used in existing works, and then propose our novel way of modeling a worker (i.e., Precision and Recall).

B. Selecting Appropriate Worker Models

• **Limitations of Accuracy and TNR.** The limitation of only using Accuracy to model a worker in [34,53] is that a single parameter cannot fully express a worker's characteristics. Some works [29,39,55] model a worker's quality with two parameters: TNR and Recall. The limitation of TNR is that it is not expressive of workers in answering multi-label tasks, because TNR =TN/(TN+FP), and in multi-label tasks, the incorrect labels are large in size and sparsely distributed in semantics. Then the crowd workers will not select most of them, making TN much larger compared with FP (Table V), resulting in a high value for TNR. As can be seen in Table VI, the TNR for all workers are very high ($\ge 85\%$), and the TNR of three workers (w_2, w_3, and w_4) are very close, making it hard to distinguish between workers.

More observations can be found in Figure 3, which shows the histograms of 141 workers' respective qualities collected

TABLE IV
CONTINGENCY TABLE OF A WORKER.

	$t = \text{Y}$	$t = \text{N}$
$v = \text{Y}$	True Positives (TP)	False Positives (FP)
$v = \text{N}$	False Negatives (FN)	True Negatives (TN)

TABLE V
EACH WORKER'S CONTINGENCY TABLE.

	TP	FP	FN	TN
w_1	3	0	4	13
w_2	5	2	2	11
w_3	3	2	4	11
w_4	3	1	0	6

TABLE VI
WORKERS' QUALITIES.

	Accuracy	Precision	Recall	TNR
w_1	80%	100%	43%	100%
w_2	80%	71%	71%	85%
w_3	70%	60%	43%	85%
w_4	90%	75%	100%	86%

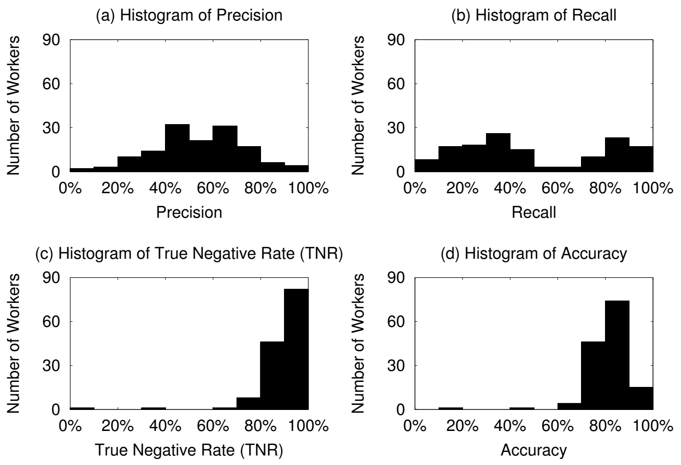


Fig. 3. Histograms of Workers' Respective Qualities.

from real-world datasets in crowdsourcing platforms (Section VII). In each figure, x-axis indicates respective quality value, and y-axis indicates the number of workers (in 141 ones) that fall in different ranges of quality values. It can be seen that the Accuracy and TNR are highly concentrated in high values, while Precision and Recall are of a wider spread and can capture the differences between workers. The reason is that Accuracy and TNR are related to TN, which dominates other three statistics: TP, FP and FN; while Precision and Recall, two parameters unrelated to TN are more expressive in modeling a worker's characteristics for multi-label tasks.

• **Our Selection: Precision and Recall.** Based on the above analysis, we model the quality of a worker using the combination of two parameters: Precision and Recall. Specifically, a worker with high Precision (e.g., w_1) means that the labels selected by the worker is likely to be correct (e.g., label `boat` for o_2); a worker with high Recall (e.g., w_4) means that the correct labels are mostly selected by the worker, which substantially indicates that a label not selected by the worker is unlikely to be correct. Moreover, a conservative worker would only select labels that the worker is certain of, yielding *high* Precision but *low* Recall (such as w_1); on the other hand, a venturous worker may select more labels, resulting in *low* Precision but *high* Recall (such as w_4).

IV. TRUTH INFERENCE IN MULTI-LABEL TASKS

This section studies the *Truth Inference Problem*, i.e., inferring the truth based on all workers' votes. We use Precision and Recall to model a worker's quality. Since the truth of each task is unknown, we need to infer each worker's model (or quality). We first propose an approach to iteratively estimate these parameters (Section IV-A), and then talk about how to speed up the computation in a real setting (Section IV-B). Note that our proposed approaches work in an unsupervised manner, i.e., it does not need any truth known in advance.

A. Iterative Method

Intuitively, the truth and workers' models have an inherent relation: if a label is selected by high-quality workers for an object, then the label is likely to be a correct label for the object; meanwhile, if a worker selects correct labels often, then the worker will be assigned with a high quality. To apply the relation, we treat the truth and worker model as two sets of parameters: (1) we denote the truth of each pair $(o_i, \ell_{i,j})$ $(1 \le i \le n,\ 1 \le j \le |L_i|)$ as $q_{i,j} \in [0,1]$, i.e., the probability that label $\ell_{i,j}$ is a correct label for o_i given all workers' votes (denoted as V, will clarify later), i.e., $q_{i,j} = \Pr(t_{i,j} = \text{Y} \mid V)$; (2) we denote the model of each worker w $(w \in \mathcal{W})$ as Precision $p_w \in [0,1]$ and Recall $r_w \in [0,1]$.

Inspired by the Expectation-Maximization (EM) framework [20], which is widely adopted by existing works [19, 28,29,32,55], we devise an iterative method. Our method first initializes the model p_w, r_w for each worker $w \in \mathcal{W}$, and then applies an iterative approach, which runs the following two steps in each iteration until convergence is attained.

Step 1. We leverage the known model p_w, r_w for each worker $w \in \mathcal{W}$ to infer the (probabilistic) truth $q_{i,j}$;

Step 2. Based on the computed truth $q_{i,j}$ in the last step, we estimate each worker w's model p_w, r_w.

Next, we talk about the **initialization**, **steps 1&2**, **convergence**, **complexity**, and analyze its **relations to EM [20]**.

• **Initialization.** A direct way is to initialize each worker $w \in \mathcal{W}$'s model is to set p_w and r_w as a fixed value $d \in [0,1]$. In experiments (Section VII) we observe that our proposed approach is robust when each worker is initialized as a decent worker, i.e., $p_w = r_w = d \ge 0.6$ for $w \in \mathcal{W}$.

• **Details of Step 1.** Based on the known (p_w, r_w) for each worker w, we infer the truth $q_{i,j} = \Pr(t_{i,j} = \text{Y} \mid V)$. To clarify, we denote $V = \{V_{i,j} \mid 1 \le i \le n, 1 \le j \le |L_i|\}$ as the vote set, where each element $V_{i,j} = \{(w,v)\}$ is a set of workers' votes for the pair $(o_i, \ell_{i,j})$, i.e., worker w votes $v \in \{\text{Y},\text{N}\}$ for $(o_i, \ell_{i,j})$. For example, from Table III we know that $V_{1,2} = \{(w_1,\text{Y}),(w_2,\text{Y}),(w_3,\text{N}),(w_4,\text{Y})\}$. Similar to [29, 34,39,53,55], we assume that the votes are given independently by workers. Then based on the Bayes' Theorem [37], we can derive the following formula

$$q_{i,j} = \frac{\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) \cdot \alpha}{\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) \cdot \alpha + \Pr(V_{i,j} \mid t_{i,j} = \text{N}) \cdot (1-\alpha)}, \quad (1)$$

where α is called *prior*, which represents the probability that a label is Y. If we already know it beforehand, we can directly set the α. However, if such information is unknown, (1) one can take a small proportion of training data (e.g., 5%) and compute the probability of labels being Y as the prior, or (2) one can compute α in an unsupervised approach, i.e., after the two steps in each iteration, α can be updated as $(\sum_{i=1}^{n}\sum_{j=1}^{|L_i|} q_{i,j})/(\sum_{i=1}^{n}|L_i|)$. We adopt the latter way, which does not require ground truth and it performs well.

Next we use the known workers' models to deduce $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$ and $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$. For worker w, its (general) worker model p_w and r_w can be represented as:

$$\begin{cases} p_w = \Pr(t_{i,j} = \text{Y} \mid v_{i,j}^{w} = \text{Y}) \ , \\ r_w = \Pr(v_{i,j}^{w} = \text{Y} \mid t_{i,j} = \text{Y}) \ . \end{cases} \quad (2)$$

(1) $\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) = \prod_{(w,v) \in V_{i,j}} \Pr(v_{i,j}^w = v \mid t_{i,j} = \text{Y})$, then if worker w votes $v = \text{Y}$, we have $\Pr(v_{i,j}^w = \text{Y} \mid t_{i,j} = \text{Y}) = r_w$ (Equation 2); otherwise, i.e., $v = \text{N}$, we have $\Pr(v_{i,j}^w = \text{N} \mid t_{i,j} = \text{Y}) = 1 - \Pr(v_{i,j}^w = \text{Y} \mid t_{i,j} = \text{Y}) = 1 - r_w$. For ease of presentation, we use the indicator function $\mathbb{1}_{\{\cdot\}}$ which returns 1 if the argument is true; 0, otherwise. For example, $\mathbb{1}_{\{2=5\}} = 0$ and $\mathbb{1}_{\{5=5\}} = 1$. Then we derive

$$\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) = \prod_{(w,v) \in V_{i,j}} (r_w)^{\mathbb{1}_{\{v=\text{Y}\}}} \cdot (1 - r_w)^{\mathbb{1}_{\{v=\text{N}\}}}. \quad (3)$$

(2) Similarly, to compute $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$, we have to know $\Pr(v_{i,j}^w = \text{Y} \mid t_{i,j} = \text{N})$, as Theorem 1 shows below.

Theorem 1: $\Pr(v_{i,j}^w = \text{Y} \mid t_{i,j} = \text{N}) = \frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w}$.

Due to space limits, interested readers can refer to technical report [43] for the details proofs of *all* theorems in the paper.

Based on Theorem 1, we can derive $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$ as

$$\prod_{(w,v) \in V_{i,j}} \left(\frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w}\right)^{\mathbb{1}_{\{v=\text{Y}\}}} \cdot \left(1 - \frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w}\right)^{\mathbb{1}_{\{v=\text{N}\}}}. \quad (4)$$

Thus with known workers' models, we can compute each $q_{i,j}$ (Equation 1) based on Equations 3 and 4.

Example 3: Suppose we have V based on Table III and workers' qualities in Table VI. We compute $q_{1,1} = \Pr(t_{1,1} = \text{Y} \mid V)$, i.e., the probability that `tree` is a true label for o_1. As w_2, w_3, w_4 vote Y and w_1 votes N, based on Equation 3, $\Pr(V_{1,1} \mid t_{1,1} = \text{Y}) = (1 - r_{w_1}) r_{w_2} r_{w_3} r_{w_4} = 0.57 * 0.71 * 0.43 * 1 = 0.174$. Similarly, based on Equation 4, $\Pr(V_{1,1} \mid t_{1,1} = \text{N}) = 0.028$. Thus from Equation 1 we get $q_{1,1} = 86\%$, i.e., `tree` is likely to be a correct label for o_1.

• **Details of Step 2.** In this step we estimate each worker's model based on the probabilistic truth computed in step 1. Note that if the truth is known, we can just count TP, FP, FN and TN for each worker. However, in last step we get the probabilistic truth $q_{i,j}$. Although we may easily decide $t_{i,j} = \text{Y(N)}$ by considering if $q_{i,j} \geq 0.5$ (or not), it cannot keep track of the probabilistic information which reflects the degree to be Y/N. Instead, we compute each worker's model by using the exact value of $q_{i,j}$. Formally, we denote D_w ($w \in \mathcal{W}$) as a set of votes given by worker w, and it contains a set of tuples $((o_i, \ell_{i,j}), v)$ representing that worker w votes $v \in \{\text{Y}, \text{N}\}$ for the pair $(o_i, \ell_{i,j})$. For example, from Table III we know $D_{w_1} = \{((o_1, \ell_{1,1}), \text{N}), \ldots, ((o_2, \ell_{2,10}), \text{Y})\}$. Then for a worker w, TP is the number of $(o_i, \ell_{i,j})$ pairs that worker w votes Y and the truth is Y, i.e., $\text{TP} = \sum_{((o_i, \ell_{i,j}), v) \in D_w} \mathbb{1}_{\{v=\text{Y}\}} \cdot \mathbb{1}_{\{t_{i,j}=\text{Y}\}}$. We can take $\mathbb{1}_{\{t_{i,j}=\text{Y}\}}$ as a random variable in $\{0, 1\}$, and compute its expected value by considering $q_{i,j}$, i.e., $\mathbb{E}[\mathbb{1}_{\{t_{i,j}=\text{Y}\}}] = q_{i,j} \cdot 1 + (1 - q_{i,j}) \cdot 0 = q_{i,j}$. Thus for worker w, we can compute $\mathbb{E}[\text{ TP }]$, and similarly $\mathbb{E}[\text{ FP }]$, $\mathbb{E}[\text{ FN }]$ as follows:

$$\begin{cases} \mathbb{E}[\text{ TP }] = \sum\limits_{((o_i, \ell_{i,j}), v) \in D_w} \mathbb{1}_{\{v=\text{Y}\}} \cdot q_{i,j}, \\ \mathbb{E}[\text{ FP }] = \sum\limits_{((o_i, \ell_{i,j}), v) \in D_w} \mathbb{1}_{\{v=\text{Y}\}} \cdot (\ 1 - q_{i,j}\ ), \\ \mathbb{E}[\text{ FN }] = \sum\limits_{((o_i, \ell_{i,j}), v) \in D_w} \mathbb{1}_{\{v=\text{N}\}} \cdot q_{i,j}. \end{cases} \quad (5)$$

Then for each worker $w \in \mathcal{W}$, we estimate its p_w and r_w as

$$p_w = \frac{\mathbb{E}[\text{ TP }]}{\mathbb{E}[\text{ TP }] + \mathbb{E}[\text{ FP }]}, \ r_w = \frac{\mathbb{E}[\text{ TP }]}{\mathbb{E}[\text{ TP }] + \mathbb{E}[\text{ FN }]}. \quad (6)$$

Note that Equation 6 does not contain TN. As we have explained in Section III-B, the very high values of TN will mislead existing worker models (e.g., Accuracy and TNR) to trust a worker, regardless of low values in precision and recall.

Example 4: In Table III we know that worker w_1 votes Y for $(o_1, \ell_{1,2})$, $(o_2, \ell_{2,2})$ and $(o_2, \ell_{2,10})$. Suppose the computed $q_{i,j}$ for those pairs are $0.9, 0.8, 0.7$, respectively, then based on Equation 5, we can estimate TP for worker w_1 as $\mathbb{E}[\text{ TP }] = 0.9 + 0.8 + 0.7 = 2.4$. Similarly we can estimate FP and FN for worker w_1, and compute p_{w_1}, r_{w_1} following Equation 6.

• **Convergence.** A typical way is to see whether or not the change of parameters (i.e., all truth and worker models) in subsequent iteration is below some predefined threshold ε (e.g., 10^{-3}). In experiments (Section VII) we observe that our proposed approach is quick to converge ($\leq$ 20 iterations).

• **Time Complexity.** In each iteration, steps 1&2 dominate the time complexity. In step 1, it enumerates all possible pairs, where for each pair $(o_i, \ell_{i,j})$, it considers all workers' answers $V_{i,j}$. Thus it is proportional to the number of all workers' answers, i.e., $\mathcal{O}(\sum_{i=1}^{n} \sum_{j=1}^{|L_i|} |V_{i,j}|)$. In step 2, it considers all workers, where for each worker w, it considers all workers' answers. Thus similar to step 1, it is also proportional to the number of all workers' answers. Suppose it takes c iterations to converge, the total time complexity is $\mathcal{O}(c \cdot \sum_{i=1}^{n} \sum_{j=1}^{|L_i|} |V_{i,j}|)$. In Section VII, we can observe that it converges very quickly ($c \leq 20$) for both real and simulated datasets.

• **Relations to EM [9].** Note that our iterative algorithm to some extend is similar to the EM algorithm [9]. In fact, most existing works that study truth inference in crowdsourcing [10,35,40,49] are based on the intuitions of EM [9] and they develop an iterative approach with two steps. However, it can be seen in experiments (Section VII) that we outperform EM a lot. Thus it is necessary to analyze our differences with EM: (1) as mentioned above, we select different worker models (i.e., precision and recall) compared with EM, making the derivations (e.g., Theorem 1) different from EM in step 1; (2) in step 2, we do not consider the very high value of TN, which misleads EM to trust a bad worker; (3) our algorithm converges quickly in both real and large simulated datasets (Section VII). In practice, we can terminate the method within a few iterations (e.g., 20). Moreover, we also develop approaches to speed up the computation (see blow).

B. Speed-Up Computation for Truth Inference

We have addressed a *static* problem, i.e., given all workers' answers, how to infer all the truth? In that case, we may not that care for the efficiency. However, assume we are faced with a situation that the input data arrives online, i.e., workers' answers come in a high velocity, and we need to instantly decide the truth. To this end, we design an incremental method, which instantly updates previously stored parameters when a new answer arrives. There are two challenges:

Challenge 1 (Storage). What parameters should be maintained for each worker w and each pair $(o_i, \ell_{i,j})$?

Challenge 2 (Update Policy). When a worker comes, which subset of parameters should be updated and how to update?

• **Solution to Challenge 1.** (1) For each worker $w \in \mathcal{W}$, we store the most related three parameters: ETP_w, EFP_w, EFN_w in database, representing the expected values of TP, FP, FN for w, respectively. (2) For each pair $(o_i, \ell_{i,j})$ ($1 \leq i \leq n, 1 \leq j \leq |L_i|$), we store $m_{i,j}^{\text{Y}}$ and $m_{i,j}^{\text{N}}$ in database, representing $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$ and $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$, respectively.

• **Solution to Challenge 2.** Suppose worker w gives the pair $(o_i, \ell_{i,j})$ a new vote v. To update the parameters, intuitively, our basic idea is to choose a small subset of *most related* parameters to update, i.e., the truth of voted pair (i.e., $q_{i,j}$) in Step 1, and all workers' models who have ever answered the voted pair (i.e., $p_{w'}$, $r_{w'}$ for $w' \in V_{i,j}$) in Step 2.
Step 1. From Equation 3 we know how to update $m^{\text{Y}}_{i,j}$ upon receiving a new vote v, i.e., if $v = \text{Y}$, then $m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot \frac{\text{ETP}_w}{\text{ETP}_w + \text{EFN}_w}$; otherwise, $m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot \frac{\text{EFN}_w}{\text{ETP}_w + \text{EFN}_w}$. We can derive similar update formula for $m^{\text{N}}_{i,j}$ from Equation 4. Then the truth can be expressed as $q_{i,j} = \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$.
Step 2. For worker w, based on Equation 5 we know that $\text{ETP}_w, \text{EFP}_w, \text{EFN}_w$ can be updated as follows: if $v = \text{Y}$, then $\text{ETP}_w = \text{ETP}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$ and $\text{EFP}_w = \text{EFP}_w + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$; otherwise, i.e., $v = N$, then $\text{EFN}_w = \text{EFN}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$. While for other workers who have answered the pair before, suppose w' ever answered v' to the pair, then if $v' = \text{Y}$, the value $\text{ETP}_{w'} = \text{ETP}_{w'} - \frac{\text{tempT}}{\text{tempT} + \text{tempN}} + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$, where tempT and tempN are $m^{\text{Y}}_{i,j}$ and $m^{\text{N}}_{i,j}$ before the above step 1. Similarly we can also have the update formulas for $\text{EFP}_{w'}$ and $\text{EFN}_{w'}$. Then $p_w = \frac{\text{ETP}_w}{\text{ETP}_w + \text{EFP}_w}$ and $r_w = \frac{\text{ETP}_w}{\text{ETP}_w + \text{EFN}_w}$.

Due to space limits, we discuss more details in the technical report [43]. Note that the designed algorithm may not achieve as high quality as the iterative approach, and the benefit is the efficiency and applicability to the scenario of a high data velocity. In practice, we can integrate the incremental approach into the iterative approach in a *delayed* manner, i.e., we run the iterative approach when receiving every 100 answers; and among the 100 answers, we only update parameters using the incremental approach. We observe that this works very well in practice and only sacrifices a little in the overall quality.

V. Incorporating Label Correlations

Since the labels of an object are not independent, in this section we study how label correlations can facilitate inferring the truth. There are two problems: (1) how to obtain the label correlations; (2) how to integrate the label correlations into our proposed method. Next we address them, respectively.

For the first problem, we can utilize existing label-correlation techniques [52,54] to generate the label correlations. In general, they can be classified into two categories: pairwise label correlations and higher order label correlations. Pairwise label correlations capture the relations between pairwise labels, which are mostly used because of its simplicity. For object o_i, the total number of pairwise label combinations is at most $|L_i|^2$. For example, the conditional dependency of two labels defines the probability that one label is correct for an object under the condition that the other label is correct. Higher oder label correlations capture the relations among subsets of labels, e.g., the co-existence probability among multiple labels, which are not frequently used mainly because of the introduced complexity in parameters. Thus we focus on pairwise label correlations in this paper, and leave higher order correlations for future work.

• **Label Correlation Function.** It can be generalized that existing works [52,54] use a small subset of training data and derive a function $\mathcal{M}(\cdot)$, which takes two labels a, b as input, and outputs a score in $[-1, 1]$ that encodes the implication of label a's correctness to label b's correctness on an object. For example, $\mathcal{M}(\texttt{sun}, \texttt{sky})$=0.9 means that label sun's correctness strongly implies label sky's correctness, i.e., if sun is correct on an object, it is highly likely that sky is also correct; while $\mathcal{M}(\texttt{happy}, \texttt{sad})$=$-0.9$ means that label happy's correctness strongly implies label sad's incorrectness, i.e., if happy is correct on an object, it is highly likely that sad is *not* correct. Recently, some open-source tools, e.g., *word2vec* [50] takes a large text corpus as input and outputs a vector for each word, which encodes the information of its adjacent words. Intuitively, the more frequent two words occur together in the text corpus, the more similar their vectors are. We can also regard each label as a word and compute the similarity (e.g., cosine similarity) between the vectors of two labels.

• **Using Label Correlations to Improve Inference.** The second problem is to facilitate addressing the truth inference problem with the known $\mathcal{M}(\cdot)$, i.e., "*Given the computed (probabilistic) truth $q_{i,j}$ ($1 \le i \le n, 1 \le j \le |L_i|$), how to refine each $q_{i,j}$ by considering label correlations (i.e., $\mathcal{M}(\cdot)$)?*"

Since $\mathcal{M}(\cdot)$ may output negative values, directly acting it on $q_{i,j}$ may contradict to the probability constraint. Inspired by the *sigmoid function* [7], i.e., $sig(x) = \frac{1}{1+e^{-x}}$ ($x \in (-\infty, +\infty)$, $sig(\cdot) \in (0, 1)$), which is a monotonic increasing function, and it has been successfully used in various models as the mapping between a real value and a probability score. So for each $q_{i,j}$ ($1 \le i \le n, 1 \le j \le |L_i|$), we consider label correlations and update $q_{i,j}$ as follows:
Step 1. Convert the probability $q_{i,j}$ to a real value $S_{i,j}$;
Step 2. Perform $\mathcal{M}(\cdot)$ on $S_{i,j}$ to derive a new $S'_{i,j}$;
Step 3. Revert $S'_{i,j}$ to a new probability $q_{i,j}$.
• **Details of Step 1.** Based on the *sigmoid* function, we set $q_{i,j} = \frac{1}{1+e^{-S_{i,j}}}$ and get $S_{i,j} = \ln \frac{q_{i,j}}{1-q_{i,j}}$. The real value $S_{i,j} \in (-\infty, +\infty)$ represents the confidence that label $\ell_{i,j}$ is correct for o_i. We can derive that $S_{i,j} \ge 0$ if $q_{i,j} \ge 0.5$, and $S_{i,j}$ increases to $+\infty$ as $q_{i,j} \to 1$; on the contrary, $S_{i,j} < 0$ if $q_{i,j} < 0.5$, and $S_{i,j}$ decreases to $-\infty$ as $q_{i,j} \to 0$.
• **Details of Step 2.** Our idea is to update $S_{i,j}$ based on the impact of related labels in o_i. Intuitively, suppose $\mathcal{M}(\ell_{i,k}, \ell_{i,j})$ is high, e.g., $\ell_{i,k}$=sun and $\ell_{i,j}$=sky, and $S_{i,k}$ is confident, i.e., if sun is likely to be a correct label for o_i, then sky is likely to be a correct label for o_i. So we update $S_{i,j}$ to $S'_{i,j}$, as follows:

$$S'_{i,j} = \alpha \cdot S_{i,j} + (1-\alpha) \cdot \sum\nolimits_{\ell_{i,k} \in RL} S_{i,k} \cdot \mathcal{M}(\ell_{i,k}, \ell_{i,j}) \ . \quad (7)$$

We can see that $S'_{i,j}$ is the weighted sum of $S_{i,j}$, and an aggregated portion of related labels (RL). For each related label $\ell_{i,k} \in RL$, it multiplies its confidence $S_{i,k}$ for o_i, and the implication of its correctness to $\ell_{i,j}$'s correctness, i.e., $\mathcal{M}(\ell_{i,k}, \ell_{i,j})$. There are two parameters in Equation 7:
(1) The weight $\alpha \in [0, 1]$ controls the impacts between itself ($S_{i,j}$) and related labels. From experiments (Section VII) we observe that it can be set as $\alpha \in [0.6, 0.7]$, which gives itself more impact, and considers related labels in the meantime.
(2) RL is a set of related labels. We can roughly set it as all labels in o_i except itself ($\ell_{i,j}$), i.e., $RL = \{\ell_{i,k} \mid 1 \le k \le |L_i| \wedge k \ne j\}$. Although we can consider other choices, e.g., based on the definition of $\mathcal{M}(\cdot)$, we can focus on the labels

$\ell_{i,k}$ whose confidence value is high (e.g., $q_{i,k} > 0.8$), we find the simply way of selecting RL performs well in practice.

• **Details of Step 3.** We revert the derived $S'_{i,j}$ to a new probability: $q_{i,j} = sig(S'_{i,j}) = 1/(1 + e^{-S'_{i,j}})$.

As the three steps require $q_{i,j}$ $(1 \le i \le n, 1 \le j \le |L_i|)$ as input, we can consider label correlations in our iterative method by running these three steps after step 1 in Section IV.

• **Time Complexity.** In label correlations, we update all truth $q_{i,j}$ with $\mathcal{M}(\cdot)$. For each $q_{i,j}$, steps 1 and 3 take constant time; in step 2 (Equation 7), it considers at most $\mathcal{O}(|L_i|)$ related labels in o_i. Thus it takes $\mathcal{O}(\sum_{i=1}^{n} |L_i|^2)$ time, which is dominated by the complexity in Section IV (i.e., $\mathcal{O}(\sum_{i=1}^{n} \sum_{j=1}^{|L_i|} |V_{i,j}|)$) if there are enough workers.

Example 5: Suppose workers' answers and qualities are known in Tables II and VI, and we take $\ell_{2,4}$ (`lake`) in o_2 as an example. In Section IV we can get $q_{2,4}$=0.26. (1) In step 1, we have $S_{2,4}$=-1.05. (2) In step 2, suppose $\mathcal{M}$(`boat`, `lake`)=0.9, and for label `boat` ($\ell_{2,10}$), similarly we get $q_{2,10}$=0.99 and $S_{2,10}$=4.6. ($q_{2,10}$ should be 1 as computed in Section IV, which makes $S_{2,10}$=$+\infty$. For illustration purpose we consider $q_{2,10} = 0.99$). To update $S_{2,4}$, for simplicity we only consider the most related label `boat`, and based on Equation 7 (α=0.7), $S'_{2,4}$=0.7$\cdot S_{2,4}$+0.3$\cdot S_{2,10}\cdot$0.9= 0.507. Finally $q_{2,4}$=$sig(S'_{2,4})$=0.62. We can see that as `boat` is confident (99%) for o_2, and the implication of `boat` to `lake` is strong (0.9), thus this will increase the probability (from 26% to 62%) that `lake` is a correct label for o_2, which makes sense.

VI. Online Task Assignment

In this section, we study the *Task Assignment Problem*: "Given the vote set V collected so far, when a worker w comes, which k objects (in all n ones) should be assigned to worker w?". To address it, we first focus on selecting a single object, and then generalize to selecting k objects.

• **Selecting a Single Object.** When a worker w comes, to decide which object should be assigned, our idea is to select the object whose ambiguity can be reduced the most *if* the object is answered by worker w. To achieve the goal, for each object o_i $(1 \le i \le n)$, we adopt the following three steps:

Step 1. We measure the object's ambiguity, denoted as $U(o_i)$;

Step 2. We calculate the *expected* ambiguity if it is answered by worker w, denoted as $\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]$;

Step 3. We compute its ambiguity reduction, denoted as

$$\Delta U(o_i) = U(o_i) - \mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]. \quad (8)$$

We select the object whose ambiguity can be reduced the most.

Details of Step 1. We need to measure o_i's ambiguity based on workers' answers (i.e., V). As V is known, which means that all inferred truth $q_{i,j}$ has been stored in the database (Figure 2). Then we can leverage them to define the ambiguity.

Definition 6 (ambiguity): We first define the ambiguity of a pair $(o_i, \ell_{i,j})$ based on Shannon Entropy [41], denoted as $U((o_i, \ell_{i,j})) = -[\ q_{i,j} \cdot \log q_{i,j} + (1 - q_{i,j}) \cdot \log(1 - q_{i,j})\ ]$. Then the ambiguity of an object o_i is defined as the sum of the ambiguity of all pairs in o_i, i.e., $U(o_i) = \sum_{j=1}^{|L_i|} U((o_i, \ell_{i,j}))$.

Note that the reason for using Shannon Entropy [41] to measure the ambiguity is its non-parametric property, which does not require any assumptions about the probability distributions. It is widely used in existing works [14,53].

Details of Step 2. To calculate the expected ambiguity of object o_i, we first study how to compute the expected ambiguity of a pair $(o_i, \ell_{i,j})$ if worker w votes for that, denoted as $\mathbb{E}[\ U((o_i, \ell_{i,j})) \mid w \text{ votes for } (o_i, \ell_{i,j})\ ]$. The challenge is that we do not know worker w's vote ($\in \{\text{Y}, \text{N}\}$) for $(o_i, \ell_{i,j})$ before o_i is assigned. To address this issue, we estimate worker w's vote based on the truth probability $q_{i,j}$ and the worker's quality. If worker w has performed tasks before, then her quality has been estimated and stored in the database already (Figure 2); otherwise, if w is a new worker, we can either initialize p_w and r_w as a fixed value (e.g., 0.6), or use the average quality of workers who have performed on the tasks before. We use the latter one since it works well in practice.

Based on worker w's quality (p_w, r_w), to calculate the expected ambiguity of a pair $(o_i, \ell_{i,j})$, there are two problems: (1) how to estimate the vote (Y/N) that worker w will give to the pair? (2) How to update $q_{i,j}$ if worker w votes Y/N for the pair? We address these two problems in Theorems 2 and 3.

Theorem 2: The probability that a worker w will vote Y for $(o_i, \ell_{i,j})$ is $\Pr(v_{i,j}^{w} = \text{Y} \mid V) = r_w \cdot q_{i,j} + \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w} \cdot (1 - q_{i,j})$.

Theorem 3: If worker w gives a new vote Y for $(o_i, \ell_{i,j})$, then $q_{i,j}$ is updated to $q_{i,j}^{\text{Y}} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \frac{\alpha\cdot(1-p_w)}{(1-\alpha)\cdot p_w}\ ]$; otherwise, if the worker's new vote is N, then $q_{i,j}$ is updated to $q_{i,j}^{\text{N}} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \big(1 - \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}\big) / (1 - r_w)\ ]$.

Based on the results in Theorems 2 and 3, we have

$$\begin{aligned} &\mathbb{E}[\ U((o_i, \ell_{i,j})) \mid w \text{ votes for } (o_i, \ell_{i,j})\ ] \\ &= U_{i,j}^{\text{Y}} \cdot \Pr(v_{i,j}^{w} = \text{Y} \mid V) + U_{i,j}^{\text{N}} \cdot \Pr(v_{i,j}^{w} = \text{N} \mid V), \text{ where} \\ &\begin{cases} U_{i,j}^{\text{Y}} = -[\ q_{i,j}^{\text{Y}} \cdot \log q_{i,j}^{\text{Y}} + (1 - q_{i,j}^{\text{Y}}) \cdot \log(1 - q_{i,j}^{\text{Y}})\ ], \\ U_{i,j}^{\text{N}} = -[\ q_{i,j}^{\text{N}} \cdot \log q_{i,j}^{\text{N}} + (1 - q_{i,j}^{\text{N}}) \cdot \log(1 - q_{i,j}^{\text{N}})\ ]. \end{cases} \end{aligned} \quad (9)$$

Note that $\Pr(v_{i,j}^{w} = \text{Y} \mid V)$, and $\Pr(v_{i,j}^{w} = \text{N} \mid V) = 1 - \Pr(v_{i,j}^{w} = \text{Y} \mid V)$ can be derived from Theorem 2, and $q_{i,j}^{\text{Y}}$, $q_{i,j}^{\text{N}}$ can be derived from Theorem 3. Intuitively, Equation 9 considers the cases that worker w will vote Y (N) for $(o_i, \ell_{i,j})$, and the updated ambiguity $U_{i,j}^{\text{Y}}$ ($U_{i,j}^{\text{N}}$) if the vote is given.

Example 6: Suppose workers' answers (V) and qualities are known in Tables III and VI. We can then compute $q_{1,1} = 0.86$ in Section IV. Suppose a new worker $w \notin \mathcal{W}$ comes and we compute both $U((o_1, \ell_{1,1}))$ and $\mathbb{E}[\ U((o_1, \ell_{1,1})) \mid w \text{ votes for } (o_1, \ell_{1,1})\ ]$. First $U((o_1, \ell_{1,1})) = -(0.86 * \log 0.86 + 0.14 * \log 0.14) = 0.4$. The new worker w's quality is estimated as the average quality: $p_w = 0.77$, $r_w = 0.64$ from Table VI. Based on Theorem 2, we get $\Pr(v_{1,1}^{w} = \text{Y} \mid V) = 0.64 * 0.86 + (\frac{0.64}{0.77} - 0.64) * 0.14 = 0.58$ and $\Pr(v_{1,1}^{w} = \text{N} \mid V) = 1 - 0.58 = 0.42$. Based on Theorem 3, we get $q_{1,1}^{\text{Y}} = 0.86/(0.86 + 0.14 * \frac{0.23}{0.77}) = 0.95$, and similarly $q_{1,1}^{\text{N}} = 0.73$. Then we can derive $U_{1,1}^{\text{Y}} = -(0.95 * \log 0.95 + 0.05 * \log 0.05) = 0.2$ and $U_{1,1}^{\text{N}} = 0.58$. Finally $\mathbb{E}[\ U((o_1, \ell_{1,1})) \mid w \text{ votes for } (o_1, \ell_{1,1})\ ] = 0.2{*}0.58 + 0.58 * 0.42 = 0.36$. Expectedly, the ambiguity of $(o_1, \ell_{1,1})$ is reduced from 0.4 to 0.36 after being voted by worker w.

Having known how to compute the expected ambiguity of a pair, we now compute the expected ambiguity of object o_i if it is answered by worker w, i.e., $\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]$. However, it is challenging to directly compute it, as it requires to enumerate all possible answers of worker w to o_i: $\{\text{Y}, \text{N}\}^{|L_i|}$, which contains $2^{|L_i|}$ items. Let $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|L_i|}\}$ denote worker w's one possible answer for o_i, where each $\sigma_j = \text{Y(N)}$ $(1 \le j \le |L_i|)$ represents that worker w

votes Y(N) for $(o_i, \ell_{i,j})$. Then for each $\sigma \in \{\text{Y},\text{N}\}^{|L_i|}$, we have to compute the ambiguity of o_i if worker w gives answer σ, denoted as U_i^σ. Based on Definition 6, it aggregates the ambiguity of all pairs in o_i considering the answer σ: $U_i^\sigma = \sum_{j=1}^{|\sigma|} -[\, q_{i,j}^{\sigma_j} \cdot \log q_{i,j}^{\sigma_j} + (1-q_{i,j}^{\sigma_j}) \cdot \log(1-q_{i,j}^{\sigma_j}) \,]$, where $q_{i,j}^{\sigma_j}$ (i.e., $q_{i,j}^{\text{Y}}$ or $q_{i,j}^{\text{N}}$) is defined in Theorem 3. If we assume that worker w will give independent votes to $(o_i, \ell_{i,j})$, then

$$\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ] = \sum_{\sigma \in \{\text{Y},\text{N}\}^{|L_i|}} U_i^\sigma \cdot \prod_{j=1}^{|L_i|} \Pr(v_{i,j}^w = \sigma_j \mid V).$$

We next prove the following theorem, which efficiently computes the expected ambiguity of an object o_i, by reducing the complexity from exponential ($2^{|L_i|}$) to linear ($|L_i|$). The idea is that we decompose the above formula into two parts, where each one considers that w votes Y (N) to a pair. Then we can verify that the expected ambiguity of the pair can be extracted from the above formula. Similarly the expected ambiguity of all pairs in o_i can be extracted and added independently.

Theorem 4: The expected ambiguity of o_i $(1 \le i \le n)$ is the sum of the expected ambiguities of all pairs in o_i, i.e.,

$$\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]$$
$$= \sum_{j=1}^{|L_i|} \mathbb{E}[\ U((o_i, \ell_{i,j})) \mid w \text{ votes for } (o_i, \ell_{i,j})\ ].$$

Despite the above theorem, note that label correlations are implicitly considered in the assignment, as it takes the derived $q_{i,j}$ (which considers label correlations in Section V) to further compute the ambiguity and estimate worker's votes.

Details of Step 3. We can compute the ambiguity reduction of object o_i: $\Delta U(o_i) = U(o_i) - \mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]$, and select a single object with the highest reduction $\Delta U(o_i)$.

• **Selecting Multiple Objects.** Next we prove the following theorem (Theorem 5), which extends our method to select k objects with the highest reduction in ambiguity.

Theorem 5: The optimal k-object combination is to select k objects with the highest reduction in ambiguity, i.e., $\Delta U(o_i)$.

• **Time Complexity.** To select top-k objects with the highest ambiguity reduction, we have to compute $\Delta U(o_i)$ for each object o_i. For an object o_i $(1 \le i \le n)$, steps 1 and 2 both take $\mathcal{O}(|L_i|)$ time, while step 3 takes constant time. Then computing $\Delta U(o_i)$ for all objects takes $\mathcal{O}(\sum_{i=1}^n |L_i|)$ time. As selecting top-k objects requires $\mathcal{O}(n)$ time (the problem of finding top-k elements in an n-array can be solved linearly [13]), the total time complexity of task assignment problem is $\mathcal{O}(\sum_{i=1}^n |L_i|)$, which is linear to #labels.

VII. Experiments

In this section, we evaluate Comlet on both real-world and simulated datasets. On real-world datasets (Sections VII-A-VII-D), we perform experiments on two crowdsourcing platforms and examine the effectiveness and efficiency of Comlet; on simulated datasets (Section VII-E), the scalability of Comlet is evaluated. Comlet is implemented in Python 2.7 and evaluated on a machine with OS Ubuntu and 16GB memory.

A. Settings

• **Two Real-World Datasets.**

Image Tagging. `Corel5k dataset` [24] contains 5000 images, tagged with 260 labels in all. We select 300 images with the most labels, and 20 labels that are most frequently tagged: {*water, sky, tree, people, grass, building, mountain, snow, flower, cloud, rock, stone, street, plane, bear, field, sand, bird, beach, boat*}. We generate 300 tasks, where each task contains an image and 20 labels.

Email Tagging. `Enron dataset` [31] contains 1700 emails, tagged with 53 labels in all. For better human readability, we filter the emails whose size is >1KB. In the remaining 837 ones, we select 300 emails with the most labels, and 20 labels that are most frequently tagged: {*forwarded emails, California energy crisis, ..., company business&strategy*}. We generate 300 tasks, where each task contains an email and 20 labels.

• **Two Crowdsourcing Platforms.**

***ChinaCrowd* [2].** ChinaCrowd is an emerging Chinese crowdsourcing platform, whose workers are mostly Chinese. We run experiments on it without quality control (i.e., no QC).

***Amazon Mechanical Turk (AMT)* [1].** AMT is a famous crowdsourcing platform. When a worker finishes a task, the task requester can give feedback to the platform on whether or not the worker's answer is approved. AMT provides each worker's historical approval rate (for past tasks) to help task requester do Quality Control. Thus we perform experiments on AMT with quality control (i.e., with QC), by setting that a worker is qualified to answer our tasks only if the worker's percentage of past approved tasks is $\ge 70\%$.

• **Collecting Workers' Answers.** We publish two datasets, i.e., image tagging and email tagging (the ground truth of the two datasets are known for evaluation purposes) on the above two platforms. So there are 4 experiments in total, where each one corresponds to a dataset on a platform. In each experiment, we assign each task to 5 workers, and pay \$0.01 for a worker upon answering a task. Thus in each experiment, we pay $300 \times 5 \times \$0.01 = \$15$ for workers. After all experiments are accomplished, we collect 4 datasets of workers' answers.

• **Comparisons.** Existing works [16,29,34,39,55] decompose a multi-label task to a set of single-label tasks and model a worker differently. For each worker model, we select one representative method and make comparisons:

***Majority Vote (MV)* [16,26].** It does not model a worker and regard the majority answer as the truth.

CDAS [34]. It uses Accuracy to model a worker, which is also used in other works [19,53]. CDAS [34] leverages workers' answers for tasks to infer each worker's Accuracy.

EM [29]. It uses both TNR and Recall to model a worker, which is also used in [39,55]. EM [29] takes workers' answers as input, and iteratively infers all workers' models.

Comlet. Our method without considering label correlations.

Comlet+. Our method with label correlations. We use a small set of *training data*, i.e., the ground truth of 5% tasks in the original dataset to derive $\mathcal{M}(\cdot)$. For two labels a and b, the output $\mathcal{M}(a, b)$ is the conditional probability that label b is correct given that label a is correct for an object. To get $\mathcal{M}(\texttt{sun}, \texttt{sky})$, for instance, we calculate the fraction of images having both labels `sun` and `sky`, over those that have `sun`.

Speed-Up. Comlet+ with speed up, as in Section IV-B.

• **Metrics.** We use different metrics to evaluate the effectiveness and efficiency when comparing different methods.

Effectiveness. We use three metrics Precision, Recall, and F1-score to measure the quality of a method. Note that many existing works [29,34,39] use Accuracy to evaluate the quality of a method. However, it does not fit to the multi-label setting,

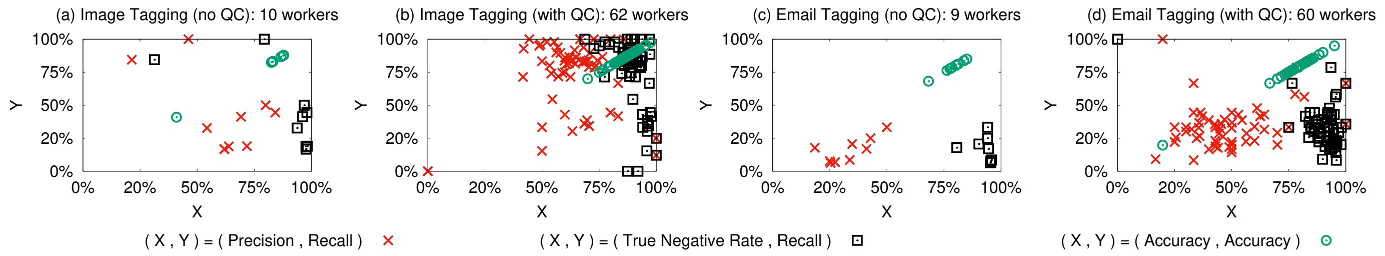


Fig. 4. Observing Workers' Real Qualities On All Collected Datasets.

because the number of incorrect labels is much higher than the correct labels, thus a method can easily attain a high Accuracy by simply not selecting any label for each task, which is useless to us. Precision and Recall are defined similarly for a method as Section III. Since these two are one-sided metrics, i.e., we can easily optimize one by sacrificing another (e.g., Recall can be optimized by selecting all labels for each task, leading to very low Precision), and their harmonic-mean, i.e., F1-score, is used [46,48,55] to measure a method's quality: $\text{F1-score} = 2 \cdot \text{Precision} \cdot \text{Recall}/(\text{Precision} + \text{Recall})$.

Efficiency. We use the execution time to evaluate efficiency.

B. Observing Workers' Real Qualities

In order to observe workers' real qualities in practice, we leverage the ground truth and use workers' answers for each dataset to compute each worker's real quality under different models. In Figure 4, each graph corresponds to the workers' real qualities in one dataset. Generally speaking, there are three ways to model a worker: (a) Accuracy [19,34,53], (b) (TNR, Recall) [29,39,55], and (c) (Precision, Recall), i.e., our model. So in each graph, we draw three points for a worker, whose coordinates (x, y) are respectively (Precision, Recall) in a '$\times$', (TNR, Recall) in a '$\boxdot$' and (Accuracy, Accuracy) in a '$\odot$'. For example, Figure 4(a) shows workers' real qualities in image tagging collected from ChinaCrowd [2] (no QC). There are 10 workers that participate in tasks, and each worker corresponds to 3 points in the graph, where each point corresponds to one of the worker's model. From Figure 4 we can observe that Accuracy and TNR are all very high for workers, even for workers with no QC, since in the 4 graphs, '$\odot$'s are located in the top right diagonal, and '$\boxdot$' are located in the very right part of x-axis. Particularly, in Section III, we have shown Figure 3, which draws the histograms of all 141 workers' qualities in the 4 datasets, and similar observations are also derived. Moreover, under our model (i.e., Precision, Recall), workers' models vary in different patterns. As can be seen, generally workers in Figure 4(a) are of *high* Precision, *low* Recall; in Figure 4(b) are of *high* Precision, *high* Recall; in Figure 4(c) are of *low* Precision, *low* Recall; in Figure 4(d) are of *mediate* Precision, *low* Recall. On one hand, this verifies the need to model a worker using Precision and Recall, as it can capture different variants of workers; on the other hand, it also brings out the challenges to achieve better results under various workers, which will be shown next.

C. Truth Inference

For the *Truth Inference Problem*, we compare our proposed methods (Comlet, Comlet+, Speed-Up) with three state-of-the-art methods (MV, CDAS, EM). We first evaluate how to set parameters, and then show the comparisons of different methods on all collected datasets.

• **Parameter Settings.** We evaluate the parameters in our iterative method, i.e., initialization, convergence (Section IV) and label correlations (Section V).

Initialization (Figure 5(a)). We evaluate the effect of different initializations on all datasets. To be specific, for each worker $w \in \mathcal{W}$, we initialize $p_w = r_w = d$ and vary $d \in [0, 1]$ in each dataset. We run enough (50) rounds to ensure it converges, and compute F1-score for different d. It can be seen in Figure 5(a) that all datasets reach its highest quality if the worker is initialized as a decent worker, i.e., $d \geq 0.6$. So we initialize each worker $w \in \mathcal{W}$ as $p_w = r_w = 0.7$.

Convergence (Figures 5(b)(c)). We study #iterations to converge. For the c-th iteration, we denote the computed truth as $q_{i,j}^{(c)}$ and worker w's model as $p_w^{(c)}$, $r_w^{(c)}$. Then the parameter change for iteration c and $c-1$ is denoted as $\tau(c)$, i.e., the sum of average differences in truth and workers' models:

$$\tau(c) = \frac{\sum_{i,j} |q_{i,j}^{(c)} - q_{i,j}^{(c-1)}|}{\sum_{i=1}^{n} |L_i|} + \frac{\sum_w (|p_w^{(c)} - p_w^{(c-1)}| + |r_w^{(c)} - r_w^{(c-1)}|)}{|\mathcal{W}|}.$$

We vary $c \in [2, 50]$ and compute $\tau(c)$ for all datasets in Figure 5(b), which shows that $\tau(c)$ becomes much smaller as c increases. We have $\tau(c) \leq 10^{-3}$ as $c \geq 20$ in all datasets. Take a further step, in Figure 5(c), we show the quality (F1-score) based on the computed truth in each iteration, which can be seen that the quality improves with the increasing c. In particular, the quality achieves highest and remains stable for different datasets as $c \geq 20$. We have also tested on large simulated data (Section VII-E), which also converge very quickly. Thus in practice, we can set #iterations (e.g., 20) to terminate the iterative inference.

Label Correlations (Figure 5(d)). We observe the effect of varying label correlation parameter α in Equation 7. As $\alpha \in [0, 1]$ measures the impact of $S_{i,j}$ (itself) and related labels, thus a larger α means more impact on $S_{i,j}$ and a smaller α means more impact on related labels. We vary $\alpha \in [0, 1]$ and observe the quality of all datasets in Figure 5(d). It can be seen that either α close to 1 or 0 results in a smaller quality, as it is not wise to totally agree with $S_{i,j}$ or related labels. We observe that the quality achieves highest as $\alpha \in [0.6, 0.7]$ for all datasets, which gives $S_{i,j}$ more impact, but at the same time considers related labels. Thus we set $\alpha = 0.7$ in practice.

• **Comparisons of Truth Inference Methods.**

Effectiveness (Figure 6 & Table VII). Figure 6 shows both one-sided quality (Precision and Recall) and two-sided quality (F1-score) for all 4 datasets. For a clear comparison, we list the statistics of different methods under the two-sided quality (i.e., F1-score) in Table VII. We can observe that (1) our proposed approaches perform much better on datasets with no QC, where Comlet and Comlet+ lead other competitors for more than 22%. As can be seen from Figures 6(a)(c),

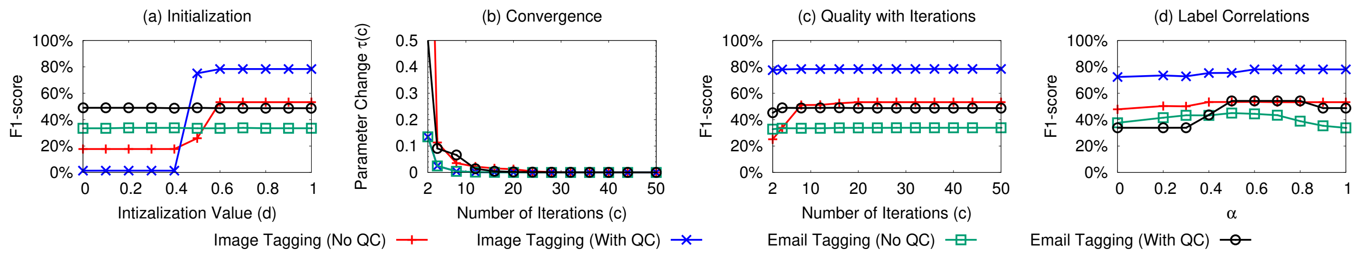


Fig. 5. Truth Inference (Parameter Settings).

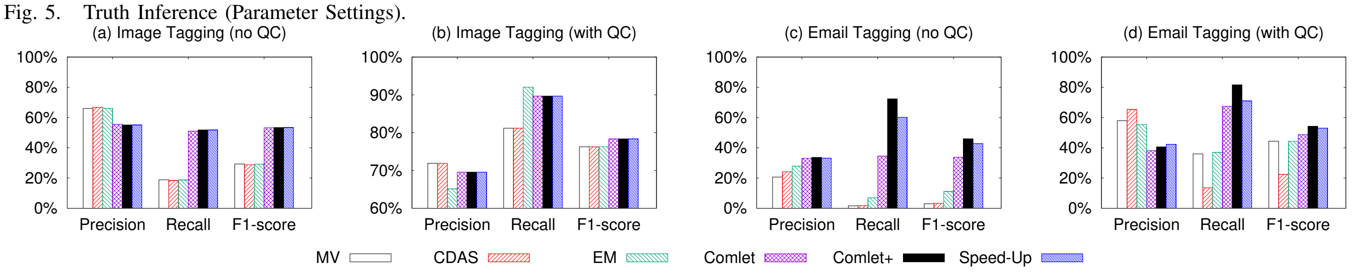


Fig. 6. Truth Inference (Effectiveness Comparisons).

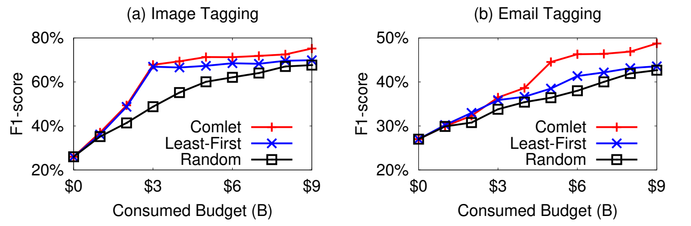


Fig. 7. Task Assignment (Effectiveness Comparisons).

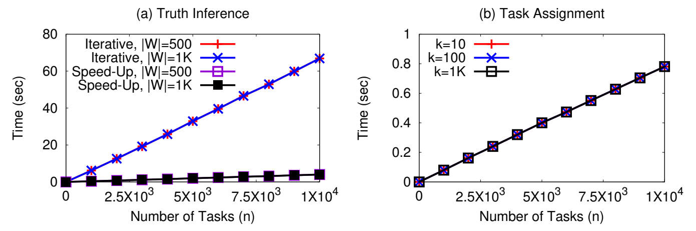


Fig. 8. Scalability on Simulated Data.

MV, CDAS and EM attain much higher Precision compared with Recall, resulting in low F1-score. The reason is that low-quality workers still have high Accuracy and TNR, thus the methods will highly trust a poor worker's N vote for a pair, and the label will be selected by the methods only if multiple Y votes are given to the pair, resulting in the limited number of selected labels. As we can observe from Figure 6, existing works have very low values in Recall, which strongly affects F1-score. However, Comlet can learn workers' diverse characteristics and make reasonable decisions, resulting in a high F1-score (with balanced Precision and Recall). (2) For datasets with QC, Comlet (Comlet+) can also outperform state-of-the-arts by 4.43% (10.03%) on email tagging dataset. Even in the case that workers are of high qualities, e.g., for image tagging, we can still outperform 2.08%. (3) By considering label correlations, Comlet+ improves Comlet a lot on email tagging: 12.16% (no QC) and 5.6% (with QC). However, there is not much improvement on image tagging. The reason is that labeling an image is much easier. Take a step further, we analyze workers' answers for image tagging, and find that most of the strong label correlations are caught by the results of Comlet (computed purely based on workers' answers), while there are only < 20 pairs that can be further benefited by considering $\mathcal{M}(\cdot)$, resulting in Comlet+'s limited improvement. However, email tagging tasks are more difficult for workers and most of the label correlations are not caught by Comlet (e.g., the high value of $\mathcal{M}(\cdot)$ on labels '*California energy crisis*' and '*company business&strategy*'). Similarly, we analyze the results of Comlet on email tagging, finding that there are > 200 pairs that can be further benefited by considering $\mathcal{M}(\cdot)$, thus Comlet+ can improve a lot compared with Comlet. (4) Speed-Up is better than Comlet, but a bit worse than Comlet+, since on one hand, it considers the label correlations; while on the other hand, the incremental approach does not perform better than the iterative approach in quality. However, Comlet+ marginally improves Speed-Up in quality, which implies that Speed-Up is also very practical.

TABLE VII
TRUTH INFERENCE COMPARISON: F1-SCORE (RUNNING TIME).

Method	*Image Tagging Dataset*		*Email Tagging Dataset*	
	no QC	*with QC*	*no QC*	*with QC*
MV	29.28% (0.2s)	76.26% (0.2s)	2.98% (0.2s)	44.36% (0.2s)
CDAS	28.79% (0.4s)	76.26% (0.4s)	3.30% (0.4s)	22.42% (0.4s)
EM	29.18% (0.9s)	76.28% (0.9s)	11.10% (0.9s)	44.32% (0.9s)
Comlet	53.25% (0.8s)	**78.36%** (0.8s)	33.79% (0.8s)	48.75% (0.8s)
Comlet+	**53.40%** (1.7s)	**78.36%** (1.7s)	**45.95%** (1.7s)	**54.35%** (1.7s)
Speed-Up	**53.40%** (0.3s)	**78.36%** (0.3s)	42.80% (0.3s)	53.03% (0.3s)

Efficiency (Table VII). Given that truth inference can be addressed off-line, we observe that all methods can be finished within 1.5s, which is efficient. MV is the most efficient since it directly derives result; EM and Comlet adopt iterative approaches which require more computations. Comlet+ considers label correlations in the iterative approach, which is the least efficient. Speed-Up is pretty efficient in real settings.

D. Task Assignment

We evaluate the effectiveness and efficiency of Comlet's task assignment (Section VI). As existing works [14,34,55] on task assignment focus on single-label tasks, which cannot be directly extended to multi-label tasks. Thus we compare with two baseline assignment strategies for multi-label tasks: (1) Random: it randomly selects k tasks; (2) Least-First: it selects k tasks that have been assigned the least times.

We use two datasets and perform experiments on AMT [1] with QC as specified in Section VII-A. For each dataset, we pay \$0.01 for a worker upon answering a task, and we set the total budget as \$9. That is, each dataset can collect 900 answers (i.e., 900×20 votes) from workers. As we have mentioned in Section I, we use the *external-questions* [5] provided by AMT [1], and embed our server on it. When a

TABLE VIII
TASK ASSIGNMENT (EFFECTIVENESS AND EFFICIENCY).

Assignment Strategy	*Image Tagging Dataset*		*Email Tagging Dataset*	
	F1-score	*Time*	*F1-score*	*Time*
Random	67.66%	10^{-5}s	42.67%	10^{-5}s
Least-First	69.89%	10^{-4}s	43.55%	10^{-4}s
Comlet	75.19%	0.025s	48.76%	0.023s

worker comes, AMT [1] passes the unique worker ID to our server, which dynamically selects 3 objects out of all n objects, and assign the generated 3 tasks to the worker. To control the effect of workers, for one dataset, we perform 3 experiments on AMT in parallel, with different assignment strategies.

Effectiveness (Figure 7 & Table VIII). We compare the effectiveness of different assignment strategies. For a fair comparison, we use Comlet's truth inference method for all assignment strategies. For each dataset, we compute the quality as the budget (denoted as B) is varied from \$0 to \$9. It can be seen that Comlet outperforms both Least-First and Random. We can also observe in Figure 7 that Least-First and Comlet perform nearly the same in the beginning ($B \in [\$0, \$3]$), and the reason is that at first the two algorithms will typically assign all 300 tasks in the dataset to workers. However, Random performs bad in the beginning, as it chooses tasks randomly, which may select some tasks multiple times and leave a few tasks not selected. As B increases, Comlet can measure the reduction in ambiguity when selecting tasks, thus gradually outperforming other competitors. In Table VIII, we give an exact comparison of F1-score as $B = \$9$ for different assignment strategies on two datasets. It shows that Comlet leads more than 5% compared with all the other competitors.

Efficiency (Table VIII). We compare the task assignment efficiency for three strategies. For each dataset, we record the worst-case assignment time in all its assignments. It can be seen that all three are efficient, and the assignment can be finished within 0.03s. Comlet is the least efficient, as it computes the ambiguity reduction for each task, and chooses k tasks with the highest reduction in ambiguity; Least-First needs to decide the tasks that have been answered with the least times, which is more expensive than Random.

E. Scalability on Simulated Data

We evaluate the scalability of truth inference and task assignment on simulated data. For each experiment, we repeat it for 1000 times and record the average time.

Scalability of Truth Inference (Figure 8(a)). We generate n tasks, where each $|L_i| = 20$. We then generate $|\mathcal{W}|$ workers, where each task is assigned to 5 randomly selected workers (from $\mathcal{W}$), and workers' answers are randomly generated. We vary $n \in [0, 10^4]$, $|\mathcal{W}| \in \{500, 1000\}$ and run both iterative and incremental truth inferences on randomly generated workers' answers. For the iterative approach, we identify convergence if the change of parameters is below $\epsilon = 10^{-3}$. It can be seen that (1) for a fixed $|\mathcal{W}|$, the time linearly increases with n; (2) when n is big enough (e.g., 10^4), the time is within 70s (3s) for iterative (incremental) method, which is efficient; (3) for iterative approach, we record that all datasets can be converged within 30 iterations, and more than 90% are within 20 iterations; (4) for a fixed n, the time does not change with varying $|\mathcal{W}|$, which conforms to the time complexity.

Scalability of Task Assignment. (Figure 8(b)). We generate n tasks, where each $|L_i| = 20$. Then we randomly generate each $q_{i,j} \in [0, 1]$, and randomly generate the coming worker w's quality $p_w \in [0, 1]$, $r_w \in [0, 1]$. We vary $n \in [0, 10^4]$, $k \in \{10, 100, 1000\}$, and run Comlet to assign k tasks for worker w, and record the time. It can be observed in Figure 8(b) that (1) for a fixed k, the time linearly increases with n, which corresponds to the complexity $\mathcal{O}(\sum_{i=1}^{n} |L_i|)$; (2) for a fixed n, the time stays the same if k varies, because after we compute each object's ambiguity reduction, we use PICK algorithm [13] to select top-k objects, which is invariant with k.

VIII. RELATED WORKS

• **Crowdsourced Data Management.** With the development in crowdsourcing [32], several platforms are built (e.g., AMT [1], CrowdFlower [4], ChinaCrowd [2]), enabling crowd workers to perform tasks. To facilitate query processing, crowdsourced databases (e.g., CrowdDB [26], Qurk [36]) are built on top of crowdsourcing platforms to support queries, e.g., join [46,48], max [45], selection [39], top-k [18], enumeration [44], crowd mining [8], visualization [12], test grading [11], advertising campaigns [30], etc.

• **Multi-Label Tasks.** In machine-learning field, multi-label tasks have been widely studied [52,54] and applied to many applications. They take features of objects as input, and train a classifier based on the training data. Different from them, we address it in crowdsourcing, i.e., the truth is inferred based on workers' answers. Furthermore, the label correlations provided by [52,54] can better facilitate our truth inference. In crowdsourcing, multi-label tasks are addressed [39,55] based on transforming each task to many *independent* single-label tasks, which will incur more latency and budget [21]. Although some recent works [23,38,39,47] focus on publishing multi-label tasks to crowdsourcing platforms, this problem is not well addressed and different characteristics of workers are not well captured. Our worker model can better capture a worker's quality in answering multi-label tasks. We also consider label correlations and online task assignment. Note that there are also some other crowdsourcing works [15,21,28] on multi-label tasks, but with different objectives, e.g., [15] studies how to build a taxonomy tree for an item, and [28] addresses to label POI (points of interest) in spatial crowdsourcing.

• **Worker Modeling.** In general, existing crowdsourcing works adopt three ways to model a worker: (1) [16,26] do not model a worker and treat each worker equally; (2) [34,53] model a worker using Accuracy; (3) [29,39,55] model a worker using True Negative Rate (TNR) and Recall. Different from them, we model a worker using Precision and Recall, which is more expressive and can capture the diverse characteristics of workers in answering multi-label tasks. Note that there are some recent works [25,35] that model each worker as a vector, which indicates a worker's ability for different domains, and they do not consider the worker's contingency table.

• **Truth Inference.** The truth inference problem has been studied in existing crowdsourcing works. Based on how a worker is modeled, the inference is done in three ways: (1) Majority Voting [16,26], which treats each worker equally; (2) [34,53] leverage Bayesian Voting to infer truth, which model Accuracy for each worker; (3) [29,55] use EM to iteratively infer truth and a worker's model (TNR and Recall). We are different from EM in that the TNR of a worker modeled by EM is highly affected by *true negatives* (TN), making workers indistinguishable, while we model a worker differently, which makes the derivations of our iterative method different. Note that there are some data integration works [22,51] that also address the truth inference problem, but their settings are different from ours since (1) they do not model workers, and (2) they cannot actively select tasks and assign to workers.

• **Task Assignment.** The task assignment problem has been studied in existing crowdsourcing works, mostly on single-label tasks [14,34,55]. Typically they measure the ambiguity of each task and assign the most ambiguous tasks to the coming worker. We are different in that we address task assignment in multi-label tasks, where an answer for a task contains the votes of multiple labels from a worker. Intuitively, we measure how much ambiguity can be reduced (rather than simply the ambiguity) of a task by considering *if* it is answered by the coming worker. We then select tasks with the highest ambiguity reduction. Although some of existing works [27] address task assignment, it assumes that the coming worker sequence has been known in advance, which does not fit to real crowdsourcing settings. There are also some other topics related to task assignment in crowdsourcing, such as worker selections [16], spatial crowdsourcing [17,28,42], etc.

IX. Conclusions and Future Work

In this paper, we study two fundamental problems about multi-label tasks. In the *Truth Inference Problem*, we use a novel worker model, and propose both iterative and incremental solutions. We also incorporate label correlations. In the *Task Assignment Problem*, when a worker comes, we select the tasks whose ambiguity can be reduced the most for the worker. We perform experiments on both real and simulated data, which verifies that our proposed approaches outperform existing methods with various workers, and are also scalable to large simulated datasets. In future work, we will study the design of UIs, and the relations among (1) the number of labels, (2) quality, (3) payments and (4) latency.

References

[1] Amazon mechanical turk. https://www.mturk.com/.
[2] Chinacrowd. http://www.chinacrowds.com.
[3] Convnet. https://code.google.com/p/cuda-convnet/.
[4] Crowdflower. http://www.crowdflower.com.
[5] External questions. http://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_ExternalQuestionArticle.html.
[6] Flickr. https://www.flickr.com/.
[7] Sigmoid function. https://en.wikipedia.org/wiki/Sigmoid_function.
[8] Y. Amsterdamer, S. B. Davidson, T. Milo, S. Novgorodov, and A. Somech. Oassis: query driven crowd mining. In *SIGMOD*, 2014.
[9] A.P.Dawid and A.M.Skene. Maximum likelihood estimation of observer error-rates using em algorithm. *Appl.Statist.*, 28(1):20–28, 1979.
[10] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, 2014.
[11] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver. How to grade a test without knowing the answers—a bayesian graphical model for adaptive crowdsourcing and aptitude testing. In *ICML*, 2012.
[12] J. P. Bigham et al. Vizwiz: nearly real-time answers to visual questions. In *UIST*, 2010.
[13] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Time bounds for selection. *JCSS*, 1973.
[14] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W.-C. Tan. Asking the right questions in crowd data sourcing. In *ICDE'12*.
[15] J. Bragg, D. S. Weld, et al. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*, 2013.
[16] C. C. Cao, J. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. *PVLDB*, 2012.
[17] Z. Chen et al. gmission: a general spatial crowdsourcing platform. *PVLDB*, 2014.
[18] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In *ICDT*, pages 225–236, 2013.
[19] G. Demartini et al. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, 2012.
[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J.R.Statist.Soc.B*, 1977.
[21] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *SIGCHI*, 2014.
[22] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2009.
[23] L. Duan et al. Separate or joint? estimation of multiple labels from crowdsourced annotations. *Expert Systems with Applications*, 2014.
[24] P. Duygulu et al. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*. 2002.
[25] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *SIGMOD*, pages 1015–1030, 2015.
[26] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, 2011.
[27] C.-J. Ho and J. W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.
[28] H. Hu et al. Crowdsourced poi labelling: Location-aware result inference and task assignment. In *ICDE*, 2016.
[29] P. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *SIGKDD Workshop*, pages 64–67, 2010.
[30] P. G. Ipeirotis and E. Gabrilovich. Quizz: targeted crowdsourcing with a billion (potential) users. In *WWW*, 2014.
[31] B. Klimt and Y. Yang. Introducing the enron corpus. In *Proceedinds of the 1st Conference on Email and Anti-Spam*, 2004.
[32] G. Li et al. Crowdsourced data management: A survey. *TKDE*, 2015.
[33] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.
[34] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 2012.
[35] F. Ma et al. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *KDD*, 2015.
[36] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, 2011.
[37] P. S. marquis de Laplace. *Théorie analytique des probabilités*. V. Courcier, 1820.
[38] S. Nowak and S. Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *MIR*, pages 557–566, 2010.
[39] A. G. Parameswaran et al. Crowdscreen: algorithms for filtering data with humans. In *SIGMOD*, 2012.
[40] V. C. Raykar et al. Learning from crowds. *JMLR*, 2010.
[41] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, Jan. 2001.
[42] H. Su, K. Zheng, J. Huang, T. Liu, H. Wang, and X. Zhou. A crowd-based route recommendation system-crowdplanner. In *ICDE*, 2014.
[43] Technical Report. http://i.cs.hku.hk/ydzheng2/Comlet-ICDE.pdf.
[44] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar. Crowdsourced enumeration queries. In *ICDE*, pages 673–684, 2013.
[45] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, 2012.
[46] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.
[47] P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *CVPR Workshop*, pages 25–32, 2010.
[48] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.
[49] J. Whitehill et al. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009.
[50] word2vec. https://code.google.com/p/word2vec/.
[51] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *TKDE*, 20(6):796–808, 2008.
[52] Y. Yu, W. Pedrycz, and D. Miao. Multi-label classification by exploiting label correlations. *Expert Systems with Applications*, 2014.
[53] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 2013.
[54] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *KDD*, pages 999–1008. ACM, 2010.
[55] Y. Zheng et al. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, 2015.

X. Appendix

In the appendix, we first show the iterative algorithm (Section X-A) and detailed incremental truth inference (Section X-B). Then we present the detailed proofs for Theorems 1- 5 (Sections X-C-X-G).

A. Iterative Algorithm

We show the detailed iterative algorithm in Algorithm 1.

Algorithm 1: *Iterative Computation Method*

```
Input: Workers' votes ($V$, $D_w$ for $w \in \mathcal{W}$), label correlations $\mathcal{M}(\cdot)$
Output: $q_{i,j}$ for $1 \le i \le n$, $1 \le j \le |L_i|$, $(p_w, r_w)$ for $w \in \mathcal{W}$
1  Initialize $(p_w, r_w)$ for $w \in \mathcal{W}$;
2  while true do
3      // Step 1: Inferring the Truth
4      for $1 \le i \le n$, $1 \le j \le |L_i|$ do
5          Compute $q_{i,j}$ using Equations 1, 3 and 4;
6      // Considering Label Correlations
7      for $1 \le i \le n$, $1 \le j \le |L_i|$ do
8          $S_{i,j} = \ln[\ q_{i,j}/(1-q_{i,j})\ ]$ ;
9          Update $S_{i,j}$ to $S'_{i,j}$ based on Equation 7;
10         $q_{i,j} = sig(\ S'_{i,j}\ ) = 1/(1+e^{-S'_{i,j}})$;
11     // Step 2: Estimating Workers' Models
12     for $w \in \mathcal{W}$ do
13         Compute $p_w$ and $r_w$ using Equations 5 and 6;
14     // Check for Convergence
15     if Converged then
16         break;
17 return $q_{i,j}$ for $1 \le i \le n$, $1 \le j \le |L_i|$, $(p_w, r_w)$ for $w \in \mathcal{W}$;
```

B. Incremental Computation

Based on the discussions in Section IV-B, we develop an incremental algorithm in Algorithm 2. The basic idea is that upon receiving a worker's answer (say worker w votes v for the pair $(o_i, \ell_{i,j})$), we only update the most important parameters, i.e., the truth of the pair and the models of workers who have answered the pair before. To be precise, we store the following parameters in order to facilitate incremental updates:
(1) for a worker $w \in \mathcal{W}$, we store three parameters: ETP_w, EFP_w, EFN_w, representing the expectations of TP, FP, FN, respectively;
(2) for a pair $(o_i, \ell_{i,j})$ ($1 \le i \le n, 1 \le j \le |L_i|$), we store $m^{\text{Y}}_{i,j}$ and $m^{\text{N}}_{i,j}$, representing $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$ and $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$ respectively.
Then if we want to compute any truth and worker's model, they can be directly returned based on the above stored parameters. For example, for the pair $(o_i, \ell_{i,j})$, we can get $q_{i,j} = \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$; for worker $w \in \mathcal{W}$, we can derive $p_w = \frac{\text{ETP}_w}{\text{ETP}_w+\text{EFP}_w}$ and $r_w = \frac{\text{ETP}_w}{\text{ETP}_w+\text{EFN}_w}$.

Algorithm 2 updates the stored parameters when it gets a new vote, i.e., a worker w votes v for a pair $(o_i, \ell_{i,j})$. It shows how the the related parameters will be updated:

Inferring the Truth (lines 2-8). In this step, as the pair $(o_i, \ell_{i,j})$ gets a new vote, we update the parameters related to $(o_i, \ell_{i,j})$, i.e., $m^{\text{Y}}_{i,j}$ and $m^{\text{N}}_{i,j}$. From Equation 3 we know how to update $m^{\text{Y}}_{i,j}$ upon receiving a new vote v, that is, if $v = \text{Y}$, then $r_w = \frac{\text{ETP}_w}{\text{ETP}_w+\text{EFN}_w}$ will be multiplied to $m^{\text{Y}}_{i,j}$; otherwise, $1 - r_w$ will be multiplied to $m^{\text{Y}}_{i,j}$. We can derive similar update formula for $m^{\text{N}}_{i,j}$ from Equation 4. These updates can be reflected in lines 3-8.

Estimating Workers' Models (lines 9-24). In this step, we update the related workers' parameters. As worker w gives a vote, then the worker's parameters (i.e., ETP_w, EFN_w, and EFP_w) should be updated. For example, based on Equation 5, we know that if receiving a new vote $v = \text{Y}$, then ETP_w will be added with $q_{i,j} = \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$. The update for $v = \text{N}$ can be similarly derived and the update of parameters for worker w correspond to lines 11-16. Other than worker w, from the first step we know that the probability $q_{i,j}$ has been updated, and this will affect the parameters for the workers who have voted for $(o_i, \ell_{i,j})$ before. In order to update those affected parameters, we have to replace the old probability with the new one. So we first use tempT and tempN to temporally store $m^{\text{Y}}_{i,j}$ and $m^{\text{N}}_{i,j}$ in line 1. Then in lines 18-24, we update each worker who has voted for $(o_i, \ell_{i,j})$ before. For example, for such a worker w', if the worker previously votes Y, then for the worker's $\text{ETP}_{w'}$, we update it by first decreasing the old probability, $\frac{\text{tempT}}{\text{tempT}+\text{tempN}}$, and then adding the new one $\frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$.

Finally we update $V_{i,j}$ by adding the new vote (line 25). Next we analyze the time complexity of Algorithm 2.

Time Complexity. For the time complexity of Algorithm 2, we only have to deal with lines 18-24 (an iteration), as other steps can be finished in constant time. For the iteration (i.e., lines 18-24), it enumerates all elements in $V_{i,j}$ and each iteration takes constant time. So the time complexity is $\mathcal{O}(|V_{i,j}|)$ (if receiving a new vote for $(o_i, \ell_{i,j})$). As $V_{i,j}$ stores all previous votes for $(o_i, \ell_{i,j})$, so the complexity is constrained by the maximum number of times a pair has been answered, which is $\mathcal{O}(|\mathcal{W}|)$. It is much more efficient compared with Algorithm 1, which costs $\mathcal{O}(c \cdot |\mathcal{W}| \cdot \sum_{i=1}^{n} |L_i|)$, especially when the number of all pairs (i.e., $\sum_{i=1}^{n} |L_i|$) is big.

C. Proof for Theorem 1

Theorem 1. $\Pr(v^w_{i,j} = Y \mid t_{i,j} = N) = \frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w}$.

Proof:

To prove it, as discussed in Section IV, we have

$$\Pr(t_{i,j} = \text{Y} \mid v^w_{i,j} = \text{Y}) = \frac{\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}) \cdot \alpha}{\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}) \cdot \alpha + \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}) \cdot (1-\alpha)}.$$

Based on Equation 2, the above formula is indeed

$$p_w = \frac{r_w \cdot \alpha}{r_w \cdot \alpha + \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}) \cdot (1-\alpha)},$$

thus $\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}) = \frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w}$. ■

Algorithm 2: *Incremental Computation Method*

```
Input: $(w, v, (o_i, \ell_{i,j}))$ (worker $w$ votes $v$ for the pair $(o_i, \ell_{i,j})$),
    $V$, $(m^{\text{Y}}_{i,j}, m^{\text{N}}_{i,j})$ for $1 \le i \le n$ and $1 \le j \le |L_i|$,
    $(\text{ETP}_w, \text{EFP}_w, \text{EFN}_w)$ for $w \in \mathcal{W}$
Output: $V$, $(m^{\text{Y}}_{i,j}, m^{\text{N}}_{i,j})$ for $1 \le i \le n$ and $1 \le j \le |L_i|$,
    $(\text{ETP}_w, \text{EFP}_w, \text{EFN}_w)$ for $w \in \mathcal{W}$
1  tempT $= m^{\text{Y}}_{i,j}$, tempN $= m^{\text{N}}_{i,j}$;
2  // (1) Inferring the Truth
3  if $v = Y$ then
4      $m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot \frac{\text{ETP}_w}{\text{ETP}_w + \text{EFP}_w}$;
5      $m^{\text{N}}_{i,j} = m^{\text{N}}_{i,j} \cdot \frac{\text{EFP}_w}{\text{ETP}_w + \text{EFN}_w}$;
6  else
7      $m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot (1 - \frac{\text{ETP}_w}{\text{ETP}_w + \text{EFP}_w})$;
8      $m^{\text{N}}_{i,j} = m^{\text{N}}_{i,j} \cdot (1 - \frac{\text{EFP}_w}{\text{ETP}_w + \text{EFN}_w})$;
9  // (2) Estimating Workers' Models
10 // (2.1) Update the parameters for worker $w$
11 if $v = Y$ then
12     $\text{ETP}_w = \text{ETP}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
13     $\text{EFP}_w = \text{EFP}_w + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
14 else
15     $\text{EFN}_w = \text{EFN}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
16 // (2.2) Update the parameters for workers who have voted the pair $(o_i, \ell_{i,j})$ before
17 for $(w', v') \in V_{i,j}$ do
18     if $v' = Y$ then
19         $\text{ETP}_{w'} = \text{ETP}_{w'} - \frac{\text{tempT}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
20         $\text{EFP}_{w'} = \text{EFP}_{w'} - \frac{\text{tempN}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
21     else
22         $\text{EFN}_{w'} = \text{EFN}_{w'} - \frac{\text{tempT}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j} + m^{\text{N}}_{i,j}}$;
23 $V_{i,j} = V_{i,j} \cup \{(w, v)\}$;
24 return $V$, $(m^{Y}_{i,j}, m^{N}_{i,j})$ for $1 \le i \le n$ and $1 \le j \le |L_i|$,
    $(ETP_w, EFP_w, EFN_w)$ for $w \in \mathcal{W}$;
```

D. *Proof for Theorem 2*

THEOREM 2. *The probability that a worker w will vote Y for $(o_i, \ell_{i,j})$ is $\Pr(v^w_{i,j} = Y \mid V) = r_w \cdot q_{i,j} + \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w} \cdot (1 - q_{i,j})$.*

Proof: Based on the Bayes' Theorem [37], considering $t_{i,j} \in \{\text{Y}, \text{N}\}$ we can derive that

$$\Pr(v^w_{i,j} = \text{Y} \mid V) = \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}, V) \cdot \Pr(t_{i,j} = \text{Y} \mid V) + \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}, V) \cdot \Pr(t_{i,j} = \text{N} \mid V).$$

Given a known $t_{i,j}$, $v^w_{i,j}$ is independent of V, thus $\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}, V) = \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}) = r_w$, $\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}, V) = \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N}) = \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}$. Then we can derive

$$\Pr(v^w_{i,j} = \text{Y} \mid V) = r_w \cdot q_{i,j} + \frac{\alpha \cdot (1-p_w) \cdot r_w}{(1-\alpha) \cdot p_w} \cdot (1 - q_{i,j}),$$

which finalizes the proof. ■

E. *Proof for Theorem 3*

THEOREM 3. *If worker w gives a new vote Y for $(o_i, \ell_{i,j})$, then $q_{i,j}$ is updated to $q^{Y}_{i,j} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \frac{\alpha\cdot(1-p_w)}{(1-\alpha)\cdot p_w}\ ]$; otherwise, if the worker's new vote is N, then $q_{i,j}$ is updated to $q^{N}_{i,j} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \big(1 - \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}\big) / (1 - r_w)\ ]$.*

Proof: If worker w gives a new vote v for $(o_i, \ell_{i,j})$, from Equations 3 and 4, we know that $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$ is updated as $\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) \cdot A$ and $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$ is updated as $\Pr(V_{i,j} \mid t_{i,j} = \text{N}) \cdot B$, where A and B are denoted as follows:

$$\begin{cases} A = (r_w)^{\mathbb{1}\{v=\text{Y}\}} \cdot (1 - r_w)^{\mathbb{1}\{v=\text{N}\}}, \\ B = \big(\frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}\big)^{\mathbb{1}\{v=\text{Y}\}} \cdot \big(1 - \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}\big)^{\mathbb{1}\{v=\text{N}\}}. \end{cases}$$

Then based on Equation 1, $q_{i,j}$ should be updated as

$$\frac{\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) \cdot A}{\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) \cdot A + \Pr(V_{i,j} \mid t_{i,j} = \text{N}) \cdot B}.$$

By dividing $A \cdot [\ \Pr(V_{i,j} \mid t_{i,j} = \text{Y}) + \Pr(V_{i,j} \mid t_{i,j} = \text{N})\ ]$ both on its numerator and denominator, we can derive that $q_{i,j}$ is updated as $q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \frac{B}{A}\ ]$.

Thus if worker w gives a new vote Y for $(o_i, \ell_{i,j})$, then $q_{i,j}$ is updated as $q^{\text{Y}}_{i,j} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \frac{\alpha\cdot(1-p_w)}{(1-\alpha)\cdot p_w}\ ]$; otherwise, if the worker's new vote is N, then $q_{i,j}$ is updated as

$q^{\text{N}}_{i,j} = q_{i,j} / [\ q_{i,j} + (1 - q_{i,j}) \cdot \big(1 - \frac{\alpha\cdot(1-p_w)\cdot r_w}{(1-\alpha)\cdot p_w}\big) / (1 - r_w)\ ]$. ■

F. *Proof for Theorem 4*

THEOREM 4. *The expected uncertainty of o_i $(1 \le i \le n)$ is the sum of the expected uncertainties of all pairs in o_i, i.e.,*

$$\mathbb{E}[\ U(o_i) \mid w \textit{ answers } o_i\ ] = \sum_{j=1}^{|L_i|} \mathbb{E}[\ U((o_i, \ell_{i,j})) \mid w \textit{ votes for } (o_i, \ell_{i,j})\ ].$$

Proof:

Given the following formula

$$\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ] = \sum_{\sigma \in \{\text{Y},\text{N}\}^{|L_i|}} U^{\sigma}_i \cdot \prod_{j=1}^{|L_i|} \Pr(v^w_{i,j} = \sigma_j \mid V).$$

We can decompose its right hand side into the sum of two parts: the first part is the sum of $2^{|L_i|-1}$ elements, considering that $\sigma_{|L_i|} = \text{Y}$ (worker w votes Y for $(o_i, \ell_{i,|L_i|})$), and the second part is the sum of the remaining $2^{|L_i|-1}$ elements, considering that $\sigma_{|L_i|} = \text{N}$ (worker w votes N for $(o_i, \ell_{i,|L_i|})$), i.e.,

$$\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ] = \sum_{\sigma' \in \{\text{Y},\text{N}\}^{|L_i|-1}} (U^{\sigma'}_i + U^{\text{Y}}_{i,|L_i|}) \cdot \Pr(v^w_{i,|L_i|} = \text{Y} \mid V) \cdot \prod_{j=1}^{|L_i|-1} \Pr(v^w_{i,j} = \sigma'_j \mid V)$$
$$+ \sum_{\sigma' \in \{\text{Y},\text{N}\}^{|L_i|-1}} (U^{\sigma'}_i + U^{\text{N}}_{i,|L_i|}) \cdot \Pr(v^w_{i,|L_i|} = \text{N} \mid V) \cdot \prod_{j=1}^{|L_i|-1} \Pr(v^w_{i,j} = \sigma'_j \mid V).$$

With a careful deduction, we can derive that

$$\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ] = \sum_{\sigma' \in \{\text{Y},\text{N}\}^{|L_i|-1}} U^{\sigma'}_i \cdot \prod_{j=1}^{|L_i|-1} \Pr(v^w_{i,j} = \sigma'_i \mid V) + \mathbb{E}[\ U((o_i, \ell_{i,|L_i|})) \mid w \text{ votes for } (o_i, \ell_{i,|L_i|})\ ].$$

The above formula shows that the expected uncertainty of the pair $(o_i, \ell_{i,|L_i|})$ can be extracted. Following the same way, the expected uncertainty of all pairs in o_i can be extracted and added independently. Thus we have

$$\begin{aligned}&\mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ]\\&=\sum\nolimits_{j=1}^{|L_i|} \mathbb{E}[\ U((o_i, \ell_{i,j})) \mid w \text{ votes for } (o_i, \ell_{i,j})\ ].\end{aligned}$$

■

G. Proof for Theorem 5

THEOREM 5. *The optimal k-object combination is to select k objects with the highest reduction in uncertainty, i.e., $\Delta U(o_i)$.*

Proof:

As a worker gives answers independently, suppose we select a fixed set of k objects (denoted as $\mathcal{T}'$) to assign for the coming worker w, i.e., $\mathcal{T}' = \{o_i\}$ and $|\mathcal{T}'| = k$, then following the proof for Theorem 4, we can similarly prove that the expected uncertainty for these k objects can be regarded as the sum of expected uncertainties of individual object, i.e.,

$$\begin{aligned}&\mathbb{E}[\ \sum\nolimits_{o_i \in \mathcal{T}'} U(o_i) \mid w \text{ answers objects in } \mathcal{T}'\ ]\\&=\sum\nolimits_{o_i \in \mathcal{T}'} \mathbb{E}[\ U(o_i) \mid w \text{ answers } o_i\ ].\end{aligned}$$

As our target is to select the optimal k-object combination, such that the uncertainty can be reduced the most. Based on the above Equation, we know that for a fixed set of objects $\mathcal{T}'$, their uncertainty reduction can be expressed as

$$\sum\nolimits_{o_i \in \mathcal{T}'} \Delta U(o_i).$$

Then in order to select the optimal k-object combination, we can treat each object independently, i.e., we compute each object o_i's uncertainty reduction $\Delta U(o_i)$, and select the top-k objects with the highest uncertainty reduction.

■