Comet: A Crowdsourcing Multi-Label Task System

Yudian Zheng †, Guoliang Li #, Reynold Cheng †, Huiqi Hu #
†The University of Hong Kong, #Tsinghua University
{ydzheng2, ckcheng}@cs.hku.hk, liguoliang@tsinghua.edu.cn, hhq11@mails.tsinghua.edu.cn

ABSTRACT

Crowdsourcing platforms, such as 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 human 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 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 Comet, a Crowdsourcing Multi-Label Task System. We have performed extensive evaluations of Comet on two real crowdsourcing platforms, i.e., AMT and ChinaCrowd. Our results show that Comet outperforms state-of-the-art approaches, and is robust under different scenarios.

1. INTRODUCTION

Crowdsourcing solutions have been proposed to solve problems that are hard for computers (e.g., sentiment analysis [27,47], entity resolution [37,39]). Consider a sentiment analysis problem: a product company, e.g., Amazon, 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 [26]. 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 [23,27,39,43,47] 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*; a person Barack Obama can be a *president*, *lawyer*, and *politician*. The multi-label tasks are not well studied in existing crowdsourcing works. Although we can transform a multi-label task to several single-label tasks, this simple approach has several weaknesses.

(1) Expensive Cost. It 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 [17], compared with single-label tasks, multi-label tasks enable six times of improvement in terms of human computation time, without sacrificing quality.

(2) Inaccurate Worker Modeling. To determine single-label tasks' results, existing solutions (e.g., [16,23,27,32,43,47]) often use the so-called *worker model* to express the behavior of the worker (e.g., the worker is a good/bad candidate for the task involved). The worker model is used to reason about the correctness (or *quality*) of the task result. However, these worker models may not be suitable for multi-label tasks. 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) [23,32,47], his quality score is $(16-2)/16 = 87.5\%$. But the extremely high score of this worker cannot reflect his 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.

(3) Negligence of Label Correlations. Existing works on single-label tasks do not consider the inherent correlations among labels. For example, in the 10 labels of Figure 1, consider one 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 (e.g., from [9,42,45]). 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. We develop a system called Comet (Figure 2), where task publisher(s) can deploy multi-label tasks on our system. Comet 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. There are two main components in Comet:

Table 1: 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}

Truth Inference. When workers submit their answers, the component infers the truth (or correct labels) of each task based on all workers' answers. Comet 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. Comet 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. Comet also 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. Comet first measures the uncertainty of each task based on the collected answers, and then estimates how much uncertainty can be reduced *if* the task is really answered by the worker. Finally the k tasks with the highest reduction in uncertainty will be assigned. As the worker's answer to the task is unknown, to compute its reduction in uncertainty, all possible answers given by the worker should be considered, which is exponential (e.g., 2^ℓ answers for ℓ labels). Comet simplifies the computation and reduces the assignment complexity from exponential to linear.

To summarize, our contributions are:
(1) We perform a comprehensive study on crowdsourcing multi-label tasks, by addressing two fundamental problems: *Truth Inference Problem* and *(Online) Task Assignment Problem* (Section 2).
(2) For the first problem, we propose an effective worker model (Section 3), and devise a method that jointly infers each task's truth and each worker's model (Section 4). We further consider how to integrate label correlations into our method (Section 5);
(3) For the second problem, we develop an effective algorithm that judiciously selects k tasks with the largest amount of uncertainty reduction for the current worker, in linear time (Section 6);
(4) We develop Comet, and use two real-world datasets to perform experiments on two crowdsourcing platforms (AMT [1] and ChinaCrowd [2]). Results show that Comet 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 Comet (Section 7).

For an overview of the paper, we define two problems: *Truth Inference Problem* and *Task Assignment Problem* in Section 2. Then we address the two problems in Sections 3-5 and Section 6, respectively. We perform experiments in Section 7, and review related works in Section 8. Finally, we conclude in Section 9.

2. THE MULTI-LABEL PROBLEM

2.1 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.*

Figure 1: An Example Task.

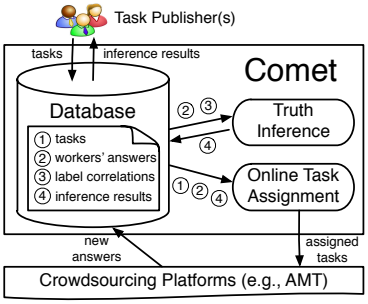

Figure 2: Comet Framework.

For example, Table 1 shows two objects o_1, o_2 and their corresponding candidate label sets L_1 and L_2. The candidate labels can be collected from user data, e.g., user image tags in Flickr [4]. 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,[1] and 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 1, 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 2 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 $\mathcal{W} = \{w_1, w_2, w_3, w_4\}, W_1 = \{w_1, w_2, w_3, w_4\}$, 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 $v_{i,j}^w = Y/N$ represents whether or not worker w selects label $\ell_{i,j}$ for o_i. To be specific, $v_{i,j}^w = Y$ (N) means that worker w selects (does not select) label $\ell_{i,j}$ for object o_i.[2] 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 1 and workers' answers in Table 2, we generate a new Table 3, which lists 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 1 and all its votes from Table 2. To make it simple, we adopt the label sequence of L_1 (L_2) as Table 1, i.e., $\ell_{1,1} =$ $\texttt{tree}$,*

[1] As there exists a 1-to-1 correspondence between tasks and objects, we use task and object interchangeably.

[2] We assume that a worker can answer a specific task at most once.

Table 2: 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 3: Truth and Votes for All (Object, Label) Pairs.

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

Table 4: Table of Notations

Notation	*Description*
o_i	the i-th object ($1 \le i \le n$)
L_i	$\{\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,\lvert L_i \rvert}\}$, or the candidate label set w.r.t. o_i
$\mathcal{T}$	$\{(o_1, L_1), (o_2, L_2), \ldots, (o_n, L_n)\}$, or the set of tasks
$\mathcal{W}$	a set of workers that have answered tasks
W_i	a set of workers that have answered o_i
$v_{i,j}^w$	Y (N) means worker w labels (does not label) $\ell_{i,j}$ for o_i
$t_{i,j}$	Y (N) means $\ell_{i,j}$ is (not) a correct label for o_i
p_w, r_w	worker w's model, or the Precision, Recall for worker w
$V_{i,j}$	a set of votes for the pair $(o_i, \ell_{i,j})$
V	$\{V_{i,j} \mid 1 \le i \le n, 1 \le j \le \lvert L_i \rvert\}$, or the vote set
$q_{i,j}$	$\Pr(t_{i,j} = \text{Y} \mid V)$, or the (probabilistic) truth of $(o_i, \ell_{i,j})$
D_w	a set of votes given by worker w
m	$m = \sum_{i=1}^{n} \lvert L_i \rvert$, or the number of all $(o_i, \ell_{i,j})$ pairs

$\ell_{1,2} = \texttt{sky}, \ldots, \ell_{1,10} = \texttt{boat}$. *For example, for the pair $(o_1, \ell_{1,2})$: its truth $t_{1,2}$ = Y, which means that label* `sky` *($\ell_{1,2}$) is a correct label for o_1; it gets 4 votes from workers, e.g., $v_{1,2}^{w_3}$ = N, which means that worker w_3 does not select* `sky` *($\ell_{1,2}$) for object o_1.*

2.2 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*. That is, given all workers' answers for objects (see Table 2), we want to infer the correct labels of objects in Table 1. We can also formulate the problem by using the data in Table 3 as below:

DEFINITION 4 (TRUTH INFERENCE PROBLEM). *Given all workers' votes, i.e., $v_{i,j}^w$ for $1 \le i \le n$, $1 \le j \le \lvert L_i \rvert$, $w \in W_i$, infer the truth of each $(o_i, \ell_{i,j})$ pair, i.e., $t_{i,j}$ for $1 \le i \le n$, $1 \le j \le \lvert L_i \rvert$.*

To address this problem, we first propose a worker model to quantify a worker's quality on a task (Section 3), and then devise an inference method to infer the truth of each task based on the worker model (Section 4). We discuss how to utilize the label correlation to further improve the inference quality (Section 5).

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, based on the current answers collected so far, called *Task Assignment Problem*.

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 6).

Table 4 summarizes the notations used in the paper.

3. WORKER MODELING

In this section, we first revisit different worker models (Section 3.1), and then point out the limitations of models used in existing works and propose our selection (Section 3.2).

To evaluate a worker's model (or quality), we should know the truth for all (object, label) pairs that the worker's votes. So in this section we assume that all truth have been known, and compare different ways of modeling a worker. In the next section (Section 4) we show how to compute each worker's model without knowing the truth (i.e., in the real-world setting).

3.1 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 5. In the table, v represents the given votes and t represents their truth, and both can be obtained from Table 3. Then for each worker, based on Table 3, 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.

EXAMPLE 3. *We show how to compute the contingency table for worker w_3 based on Table 3. As worker w_3 has answered 2 objects, by giving votes Y for 5 pairs and votes N for 15 pairs. So TP+FP=5 and FN+TN=15. Among the five Y votes:* `tree`, `flower` *for o_1, and* `lake`, `sun`, `building` *for o_2 (i.e., $v_{1,1}^{w_3} = v_{1,8}^{w_3} = v_{2,4}^{w_3} = v_{2,6}^{w_3} = v_{2,7}^{w_3}$ = Y), the truth of 3 labels are Y (i.e., $t_{1,1} = t_{2,4} = t_{2,6}$ = Y), thus TP=3, FP=2. Similarly we can get FN=4 and TN=11 for worker w_3. We list all four workers' contingency-table values in Table 6. Note that as worker w_4 only answers o_1, so its contingency table is computed based on worker w_4's votes for o_1.*

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), defined as follows:

- **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 (i.e., its truth is Y):
  $\Pr(\ t = \text{Y} \mid v = \text{Y}\ ) = \frac{\text{TP}}{\text{TP+FP}}$.
- **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 6, the worker's different quality parameters can be derived in Table 7. For these parameters, existing works [23,27,32,43,46,47] either use only one

Table 5: 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 6: 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 7: 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%

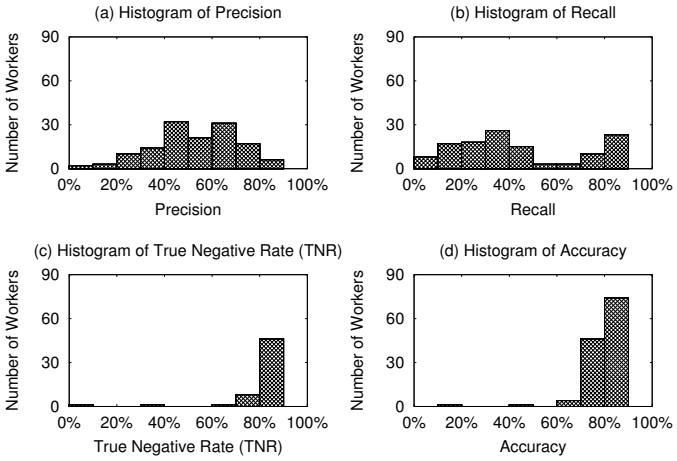

Figure 3: Histograms of Workers' Respective Qualities.

or a combination of them to model a worker's quality: Accuracy is used in [27,43,46] to model a worker's quality, while [23,32,47] use a combination of TNR and Recall to model a worker's quality. In the following section (Section 3.2), 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).

3.2 Selecting Worker Model

Limitations of Accuracy. The limitation of only using Accuracy to model a worker in [27,43,46] is that a single parameter cannot fully express a worker's characteristics. For example, in Table 2 we know that three workers answer o_2, and label `boat` gets one Y vote from w_1 and two N votes from w_2, w_3. Consider the three workers' Accuracy in Table 7, as w_1 and w_2 (with the same Accuracy) give contradictory votes, while w_3 (with Accuracy 70%) votes N, so label `boat` will be decided as not in o_2 if we only consider Accuracy of each worker. However, we can see that w_1 is of a high Precision (100%), meaning that the worker's Y vote strongly implies the truth to be Y, i.e., label `boat` is highly likely to be a correct label for o_2. Moreover, for w_2 and w_3, as their Recall are not so high (especially for w_3, with Recall 43%), meaning that the correct labels are likely to be missed by them. If we use both Precision and Recall (Section 4), we can derive that o_2 has label `boat`, which corresponds to the truth.

Limitations of TNR. Some works [23,32,47] 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 6), resulting in a high value for TNR. As can be seen in Table 7, the TNR for all workers are very high ($\geq$ 85%), and the TNR of three workers (w_2, w_3, and w_4) are very close, making it hard to distinguish between different workers. However, if we take a close look at these three workers, w_2 and w_4 are of relatively high Precision and Recall, while the Precision and Recall for w_3 are relatively low (both $\leq$ 60%), so TNR cannot express the wide spread of a worker.

More observations can be found in Figure 3, which shows the histograms of 141 workers' respective qualities collected from real-world datasets in crowdsourcing platforms (Section 7). 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 and of a low spread, 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 (also shown in Table 6); while Precision and Recall, two parameters unrelated to TN are more expressive in modeling a worker.

Our Selection: Precision and Recall. Based on the above analysis, we model the quality of a worker using two parameters: Precision and Recall, and the combination has not been considered in existing crowdsourcing works. 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).

4. ITERATIVE TRUTH INFERENCE METHOD

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 introduce our general principle, which captures the inherent relation between the truth and workers' models (Section 4.1). We then apply the principle to developing an iterative method that solves the problem (Section 4.2).

4.1 General Principle

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 principle, we treat the truth and worker model as two sets of parameters, and compute them in an iterative way. To be specific, we denote the truth of each $(o_i, \ell_{i,j})$ ($1 \leq i \leq n$, $1 \leq j \leq |L_i|$) as $q_{i,j} \in [0, 1]$, which is 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., $q_{i,j} = \Pr(t_{i,j} = \text{Y} \mid V)$; we denote the model of each worker w ($w \in \mathcal{W}$) as Precision $p_w \in [0, 1]$ and Recall $r_w \in [0, 1]$. In each iteration, we devise two steps:

Step 1: we assume that the model p_w, r_w for each worker w is known, and infer the (probabilistic) truth $q_{i,j}$ for each $(o_i, \ell_{i,j})$ pair, which is called *Inferring the Truth*;

Step 2: based on the computed truth $q_{i,j}$ for each $(o_i, \ell_{i,j})$ pair, we estimate each worker w's model p_w, r_w, which is called *Estimating Workers' Models*.

4.2 Iterative Method

4.2.1 Inferring the Truth

Based on the known (p_w, r_w) for each worker w, we infer the truth $q_{i,j} = \Pr(t_{i,j} = \text{Y} \mid V)$ for each $(o_i, \ell_{i,j})$ pair (note that as $t_{i,j}$ = Y/N, we know that $\Pr(t_{i,j} = \text{N} \mid V) = 1 - q_{i,j}$). 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 for $(o_i, \ell_{i,j})$, where v is Y or N. For example, from Table 3 we know that $V_{1,2} = \{(w_1, \text{Y}), (w_2, \text{Y}), (w_3, \text{N}), (w_4, \text{Y})\}$.

We assume that the votes are given independently by workers (this is a typical assumption adopted in existing works [23,27,32, 43,46,47]), then $t_{i,j}$ is only related to $V_{i,j}$. Based on the Bayes' Theorem [8], we get

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

Here $\Pr(t_{i,j} = \text{Y})$ and $\Pr(t_{i,j} = \text{N})$ are called *priors*. We can set their values if we have the information. Otherwise, similar to [27, 32], we assume uniform priors here, i.e., $\Pr(t_{i,j} = \text{Y}) = \Pr(t_{i,j} = \text{N}) = 0.5$. Then

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

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:

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

(1) To compute $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$, we have

$$\Pr(V_{i,j} \mid t_{i,j} = \text{Y}) = \prod_{(w,v) \in V_{i,j}} \Pr(v^w_{i,j} = v \mid t_{i,j} = \text{Y}).$$

That is, $\Pr(V_{i,j} \mid t_{i,j} = \text{Y})$ depends on workers' votes. If worker w votes $v = \text{Y}$, then $\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}) = r_w$ (Equation 2); otherwise, if $v = \text{N}$, then $\Pr(v^w_{i,j} = \text{N} \mid t_{i,j} = \text{Y}) = 1 - \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{Y}) = 1 - r_w$. 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 the following equation

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

(2) To compute $\Pr(V_{i,j} \mid t_{i,j} = \text{N})$, similarly we have

$$\Pr(V_{i,j} \mid t_{i,j} = \text{N}) = \prod_{(w,v) \in V_{i,j}} \Pr(v^w_{i,j} = v \mid t_{i,j} = \text{N}).$$

Note that as $\Pr(v^w_{i,j} = \text{N} \mid t_{i,j} = \text{N}) = 1 - \Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N})$, we only need to deduce $\Pr(v^w_{i,j} = \text{Y} \mid t_{i,j} = \text{N})$, which can be proved in Theorem 1.

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

Due to space constraint, interested readers can refer to the technical report [6] for the detailed proof. We can then derive

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

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

EXAMPLE 4. *Suppose we have V based on Table 3 and workers' qualities in Table 7. We take $(o_1, \ell_{1,1})$: label* `tree` *$(\ell_{1,1})$ in o_1 as an example and compute $q_{1,1} = \Pr(t_{1,1} = Y \mid V)$. As w_2, w_3, w_4 vote Y and w_1 votes N, based on Equation 3, $\Pr(V_{1,1} \mid t_{1,1} = Y) = (1 - r_{w_1}) r_{w_2} r_{w_3} r_{w_4} = 0.57 * 0.71 * 0.43 * 1 = 0.174$. Similarly, based on Equation 4, $\Pr(V_{1,1} \mid t_{1,1} = N) = 0.028$. Thus from Equation 1 we get $q_{1,1} = 86\%$, i.e.,* `tree` *is likely to be a correct label for o_1.*

4.2.2 Estimating Workers' Models

Next we estimate each worker's model based on the probabilistic truth computed in the last step. Note that if the truth is clearly known, we can derive all workers' contingency tables (e.g., Table 6) by simply counting TP, FP, FN and TN for each worker. However, in last step we get the probabilistic truth $q_{i,j}$ for each pair $(o_i, \ell_{i,j})$. Although we may easily decide $t_{i,j} = \text{Y(N)}$ by considering if $q_{i,j} \ge 0.5$ (or not), it cannot keep track of the probabilistic information which reflects the degree to be Y/N. Instead, we compute TP, FP, FN and TN for each worker 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 for the pair $(o_i, \ell_{i,j})$, where v is Y or N. For example, from Table 3 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., $\text{TP} = \sum_{((o_i,\ell_{i,j}),v) \in D_w} \mathbb{1}_{\{v=\text{Y}\}} \cdot \mathbb{1}_{\{t_{i,j}=\text{Y}\}}$. As $\mathbb{1}_{\{t_{i,j}=\text{Y}\}}$ is not clearly known, we can treat it as a random variable $\mathbb{1}_{\{t_{i,j}=\text{Y}\}} \in \{0, 1\}$ 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 $\mathbb{E}[\text{ FP }]$, $\mathbb{E}[\text{ FN }]$, and $\mathbb{E}[\text{ TN }]$ as follows:

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

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

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

EXAMPLE 5. *In Table 3 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, FN, TN for worker w_1 and compute p_{w_1}, r_{w_1} following Equation 6.*

4.2.3 Iterative Computation Algorithm

We design an iterative method in Algorithm 1. It takes all workers' votes as input, and outputs all probabilistic truth and workers' models. With the initialized workers' models in line 1, it adopts an iterative approach. In each iteration (lines 3-16), the two steps (i.e., inferring the truth and estimating workers' models) are run respectively in lines 3-5 and lines 11-13 (we address label correlations, i.e., lines 6-10 in next section). Then it decides whether to terminate by checking convergence in lines 14-16. Next we address the **initialization**, **convergence** and **time complexity**, respectively.

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 *// 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 *// 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* $w \in \mathcal{W}$;

Initialization. To initialize all workers' models, a direct way is to assign p_w and r_w with a fixed value $d \in [0,1]$ for each worker w. In experiments (Section 7) 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 $w \in \mathcal{W}$.

Convergence. To check the 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 we observe that our proposed approach is quick to converge (≤ 20 iterations). We will discuss more in Section 7.

Time Complexity. In each iteration, there are three main parts: (1) *inferring the truth* (lines 3-5), which requires to enumerate all $m = \sum_{i=1}^{n} |L_i|$ pairs, where for each pair $(o_i, \ell_{i,j})$, it considers all tuples in $V_{i,j}$. As $V_{i,j}$ contains at most $|\mathcal{W}|$ tuples (all workers give votes for the pair), the complexity is $\mathcal{O}(m \cdot |\mathcal{W}|)$. (2) *estimating workers' models* (lines 11-13), which requires to enumerate $|\mathcal{W}|$ workers, where for each worker w, it considers all tuples in D_w. As a worker can at most vote for all m pairs, then $|D_w| \le m$, and this part takes $\mathcal{O}(m \cdot |\mathcal{W}|)$ time. (3) *Convergence* (lines 14-16), which requires to check all parameters, so the complexity is $\mathcal{O}(m + |\mathcal{W}|)$. To summarize, each iteration takes $\mathcal{O}(m \cdot |\mathcal{W}|)$ time, and suppose it takes c iterations to converge, the total time complexity is $\mathcal{O}(c \cdot m \cdot |\mathcal{W}|)$. In Section 7, we can observe that it converges very quickly ($c \le 20$) for various real datasets.

4.2.4 Incremental Computation

As the input data arrives online, i.e., workers' answers come in a high velocity, we need to instantly decide the truth. To this end, we design an incremental method that instantly updates previously stored parameters when a new answer arrives. The basic idea is that upon receiving a worker's new answer, we only choose a small subset of *related* parameters to update, e.g., the truth of voted pairs, and all workers' models who have ever answered the voted pairs. Interested readers can refer to the technical report [6] for the details.

5. 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 sub-problems. The first is how to obtain the label correlations. The second is how to integrate the label correlations into our proposed method. Next we address them respectively.

We can utilize existing label-correlation techniques [9,42,45] to generate the label correlations and regard them as *prior* input to our problem. Generally 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 high 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 [9,42,45] 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* [40] takes a large text corpus as input and outputs a vector for each word, encoding the information of its adjacent words. Intuitively, the more frequent two words occur together in 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) of the two labels' vectors.

Using Label Correlations to Improve Inference. Our problem is how 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 [5], 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: we first convert the probability $q_{i,j}$ to a real value $S_{i,j}$;
Step 2: we act $\mathcal{M}(\cdot)$ on $S_{i,j}$ to derive a new $S'_{i,j}$;
Step 3: we revert $S'_{i,j}$ to a new probability $q_{i,j}$.

In the first step, 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 $q_{i,j} \to 1$; $S_{i,j} < 0$ if $q_{i,j} < 0.5$, and $S_{i,j}$ decreases to $-\infty$ as $q_{i,j} \to 0$.

In the second step, 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 multiples 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 $\alpha \in [0, 1]$, which controls the impacts between itself ($S_{i,j}$) and related labels. From experiments (Section 7) we observe that it can be set as $\alpha \in [0.6, 0.7]$, which gives itself more impact, but at the same time considers related labels.
(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., $RL = \{\ell_{i,k} \mid 1 \le k \le |L_i| \wedge k \ne j\}$. However, by considering the definition of $\mathcal{M}(\cdot)$, we focus on the labels $\ell_{i,k}$ whose confidence value is high (e.g., $q_{i,k} > 0.8$, i.e., $S_{i,k} = \ln \frac{q_{i,k}}{1-q_{i,k}} > 1.4$) and the implication is strong (e.g., $|\mathcal{M}(\ell_{i,k}, \ell_{i,j})| \ge 0.8$), then $RL = \{\ell_{i,k} \mid 1 \le k \le |L_i| \wedge k \ne j \wedge S_{i,k} > 1.4 \wedge |\mathcal{M}(\ell_{i,k}, \ell_{i,j})| \ge 0.8\}$, which removes a lot of unnecessary computations from not strongly related labels.

In the third step, 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 *inferring the truth*, which corresponds to lines 6-10 in Algorithm 1.

Time Complexity. In each iteration of Algorithm 1, 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. Let $e = \max_{1 \le i \le n} |L_i|$, then it takes $\mathcal{O}(m \cdot e)$ for all pairs, which is dominated by other parts in last section (i.e., $\mathcal{O}(m \cdot |\mathcal{W}|)$) if there are enough workers.

EXAMPLE 6. *Suppose workers' answers and qualities are known in Tables 2 and 7, and we take $\ell_{2,4}$ (*`lake`*) in o_2 as an example. In Section 4 we can get $q_{2,4}$=0.26. Then it is converted to $S_{2,4}$=−1.05. 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.*[3] *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·$S_{2,4}$+0.3·$S_{2,10}$·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), this increases the probability (from 26% to 62%) that* `lake` *is correct for o_2.*

6. 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?".

In selecting k objects, our basic idea is to estimate how much uncertainty can be reduced for each object, by considering *if* the task will be answered by w. Then we select optimal k-object combination with the highest uncertainty reduction. However, to achieve the goal, there are two challenges here: (1) to select k out of n objects, we have to consider all $\binom{n}{k}$ combinations, which is exponential; (2) even for a single object o_i, to consider how worker w may answer it, there are still $2^{|L_i|}$ possible cases, which is exponential.

To address it, in the following, we first focus on selecting a single object, and then generalize to selecting multiple (e.g., k) objects.

Selecting a Single Object. When a worker w comes, to decide which object should be assigned to w, our idea is to select the object whose uncertainty 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 uncertainty, denoted as $U(o_i)$;
Step 2: we calculate the *expected* uncertainty 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 uncertainty reduction, denoted as

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

We select the object whose uncertainty is reduced the most.

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

DEFINITION 6 (UNCERTAINTY). *We first define the uncertainty of a pair $(o_i, \ell_{i,j})$ based on Shannon Entropy [35], 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 uncertainty of an object o_i is defined as the sum of the uncertainty 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 [35] to measure the uncertainty is its non-parametric property, which does not require any assumptions about the probability distributions. It is widely used in existing works [11,43,44].

In the second step, to calculate the expected uncertainty of object o_i by considering if it is answered by worker w, we first study how to compute the expected uncertainty 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 (Y or 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 the worker's quality has been estimated and stored in the database already (Figure 2); otherwise, if w is a new worker, we can use the average quality (i.e., precision and recall) of workers that have performed on the tasks, which can be derived from the database.

Formally, we denote worker w's quality as p_w, r_w. To calculate the expected uncertainty of a pair $(o_i, \ell_{i,j})$, we (1) estimate the vote (Y/N) that worker w will give to the pair, and (2) estimate how the truth $q_{i,j}$ is updated 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 $\Pr(v^w_{i,j} = Y \mid V) = r_w \cdot q_{i,j} + (\frac{r_w}{p_w} - r_w)(1 - q_{i,j})$.*

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

Due to space limits, interested readers can refer to technical report [6] for the detailed proofs of Theorems 2 and 3. Then we have

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

Note that $\Pr(v^w_{i,j} = \mathrm{Y} \mid V)$, and $\Pr(v^w_{i,j} = \mathrm{N} \mid V) = 1 - \Pr(v^w_{i,j} = \mathrm{Y} \mid V)$ can be derived from Theorem 2, and $q^{\mathrm{Y}}_{i,j}$, $q^{\mathrm{N}}_{i,j}$ 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 uncertainty $U^{\mathrm{Y}}_{i,j}$ ($U^{\mathrm{N}}_{i,j}$) if the vote is given.

[3] $q_{2,10}$ should be 1 (as $p_{w_1} = 1$) following Algorithm 1, which makes $S_{2,10} = +\infty$. For illustration purpose we consider $q_{2,10} = 0.99$.

EXAMPLE 7. *Suppose workers' answers (V) and qualities are known in Tables 3 and 7. We can compute $q_{1,1} = 0.86$ based on Algorithm 1. Suppose a new worker $w \notin \mathcal{W}$ comes and we compute both $U((o_1, \ell_{1,1}))$ and $\mathbb{E}[\, U((o_1, \ell_{1,1})) \mid w$ 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 7. Based on Theorem 2, we get $\Pr(v^w_{1,1} = Y \mid V) = 0.64 * 0.86 + (\frac{0.64}{0.77} - 0.64) * 0.14 = 0.58$ and $\Pr(v^w_{1,1} = N \mid V) = 1 - 0.58 = 0.42$. Based on Theorem 3 we get $q^Y_{1,1} = 0.86/(0.86 + 0.14 * \frac{0.23}{0.77}) = 0.95$, and similarly $q^N_{1,1} = 0.73$. Then we can derive $U^Y_{1,1} = -(0.95 * \log 0.95 + 0.05 * \log 0.05) = 0.2$ and $U^N_{1,1} = 0.58$. Finally $\mathbb{E}[\, U((o_1, \ell_{1,1})) \mid w$ votes for $(o_1, \ell_{1,1})\,] = 0.2 * 0.58 + 0.58 * 0.42 = 0.36$. Expectedly, the uncertainty 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 uncertainty of a pair, we now compute the expected uncertainty of object o_i if it is answered by worker w, i.e., $\mathbb{E}[\, U(o_i) \mid w$ 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: $\{\mathrm{Y}, \mathrm{N}\}^{|L_i|}$, containing $2^{|L_i|}$ items. Let $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|L_i|}\}$ denote worker w's one possible answer for o_i, where each $\sigma_j = \mathrm{Y(N)}$ $(1 \le j \le |L_i|)$ represents that worker w votes Y(N) for $(o_i, \ell_{i,j})$. Then for each $\sigma \in \{\mathrm{Y}, \mathrm{N}\}^{|L_i|}$, we have to compute the uncertainty of o_i if worker w gives answer σ, denoted as U^σ_i. Based on Definition 6, it aggregates the uncertainty of all pairs in o_i considering the answer σ: $U^\sigma_i = \sum_{j=1}^{|\sigma|} -[\, q^{\sigma_j}_{i,j} \cdot \log q^{\sigma_j}_{i,j} + (1 - q^{\sigma_j}_{i,j}) \cdot \log(1 - q^{\sigma_j}_{i,j})\,]$, where $q^{\sigma_j}_{i,j}$ (i.e., $q^{\mathrm{Y}}_{i,j}$ or $q^{\mathrm{N}}_{i,j}$) is defined in Theorem 3. If we assume that worker w will give independent votes to $(o_i, \ell_{i,j})$, then

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

We next prove the following theorem, which efficiently computes the expected uncertainty of an object o_i, reducing the complexity from exponential ($2^{|L_i|}$) to linear ($|L_i|$). The basic idea is that if we decompose the above formula as two parts, where each part considers that w votes Y (N) to a pair (e.g., $(o_i, \ell_{i,1})$), then we can verify that the expected uncertainty of the pair can be extracted from the above formula. Similarly the expected uncertainty of all pairs in o_i can be extracted and added independently. Interested readers can refer to technical report [6] for detailed proof.

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.,*

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

In the third step, we can compute the uncertainty reduction of object o_i: $\Delta U(o_i) = U(o_i) - \mathbb{E}[\, U(o_i) \mid w$ answers $o_i\,]$ and select a single object with the highest reduction $\Delta U(o_i)$. Note that label correlations are implicitly considered in the assignment, as it takes the derived $q_{i,j}$ (which considers label correlations in Algorithm 1) to further compute the uncertainty and estimate worker's votes.

Selecting Multiple Objects. When a worker comes, existing crowdsourcing platforms such as AMT [1] can support to batch multiple tasks in a HIT (Human Intelligence Task), and assign for the coming worker. Similar to the proof in Theorem 4, we can easily extend our method to select k objects in Theorem 5, by retrieving top-k objects with highest reduction in uncertainty. Interested readers can refer to technical report [6] for detailed proof.

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

Time Complexity. To select top-k objects with highest uncertainty reduction, we have to compute $\Delta U(o_i)$ for each object o_i. For an object o_i $(1 \le i \le n)$, step 1 takes $\mathcal{O}(|L_i|)$ time (Definition 6); step 2 also takes $\mathcal{O}(|L_i|)$ time (Theorem 4); step 3 takes constant time (Equation 8). Then computing $\Delta U(o_i)$ for all objects takes $\mathcal{O}(m)$ time, where $m = \sum_{i=1}^{n} |L_i|$. 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 using the PICK algorithm [10]), the total time complexity of task assignment problem is $\mathcal{O}(m)$.

7. EXPERIMENTAL EVALUATIONS

In this section, we evaluate Comet both on real-world and simulated datasets. On real-world datasets (Sections 7.1-7.4), we perform experiments on two crowdsourcing platforms and examine the effectiveness and efficiency of Comet; on simulated data (Section 7.5), the scalability of Comet is evaluated. Comet is implemented in Python 2.7 on an 8GB memory Windows 7 machine.

7.1 Settings

Two Real-World Datasets

Image Tagging. `Corel5k dataset` [19] 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` [24] 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 email, arrangement, company business&strategy, new text&forwarded material, attachment, personal context, document checking, employment arrangement, secret message, internal project, California energy crisis, regulation, internal company operation, political influence, purely personal, point to url, alliance, internal company policy, empty message, current company image}. We generate 300 tasks, where each task contains an email (with contents in it) and 20 labels.

Two Crowdsourcing Platforms

ChinaCrowd [2]. ChinaCrowd is an emerging Chinese crowdsourcing platform, whose workers are mostly Chinese people. We perform 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 publisher 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 publisher 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 historical approval rate 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 an experiment, we assign each task to 5 workers, and pay \$0.01 for a

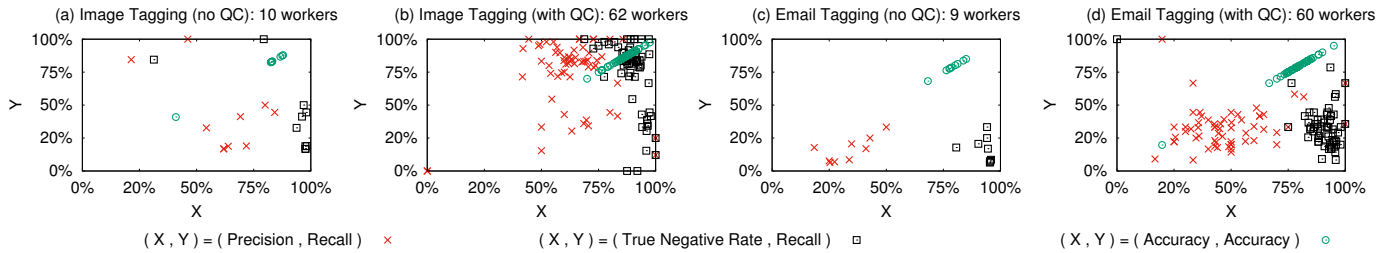

Figure 4: Observing Workers' Real Qualities On All Collected Datasets.

worker upon answering a task. Thus for 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 [13,23,27,31,32,47] model a worker differently. For each worker model, we select one representative:

Majority Vote (MV) [13,31]. It does not model a worker and treat each worker equally. Given the collected 5 votes for a pair, it decides to return Y/N that attains more votes (or ≥ 3 votes).

CDAS [27]. It uses Accuracy to model a worker, which is also used in [16,43]. CDAS [27] leverages workers' answers for tasks to infer each worker's Accuracy.

EM [23]: It uses both TNR and Recall to model a worker, which is also used in [32,47]. EM [23] takes workers' answers as input, and iteratively infer all workers' models.

Comet: Our method without considering label correlations.

Comet+: Our method that considers label correlations. For 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`.

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[4]. Precision and Recall are defined similarly for a method as Section 3. 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 often used [37–39,47] to measure a method's quality:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Efficiency. We use the execution time to evaluate efficiency.

7.2 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 [16,27,43], (b) TNR+Recall [23,32,47], and (c) Precision+Recall (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 (1) AMT [1] has more workers participating in tasks than ChinaCrowd [2], and this is reasonable as ChinaCrowd [2] is a new platform, while AMT has a lot more active workers. (2) Accuracy and TNR are all very high for workers, even for workers with no QC: 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 3, we have shown Figure 3, which draws the histograms of all 141 workers' qualities in the 4 datasets, and the same observations can be derived. (3) Under our model (i.e., Precision, Recall), workers' models vary in different settings. 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.

7.3 Truth Inference

For the *Truth Inference Problem*, we compare our proposed methods (Comet, Comet+) 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 4) and label correlations (Section 5).

Initialization. Figure 5(a) shows the quality for 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. In Figures 5(b)(c), we study the number of iterations to converge. We identify convergence if the change of parameters (all truth and worker models) between subsequent iteration is below some predefined threshold ε. 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:

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

[4] Note that many existing works [23,27,32] use Accuracy to evaluate the quality of a method. However, it does not fit to the multi-label setting, 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.

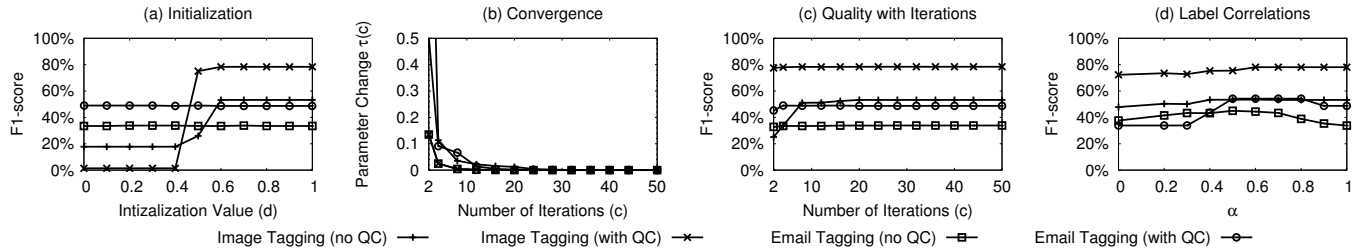

Figure 5: Truth Inference (Parameter Settings).

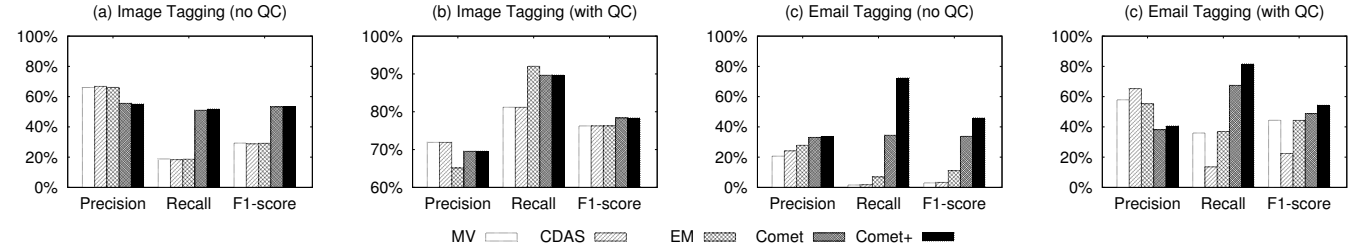

Figure 6: Truth Inference (Effectiveness Comparisons).

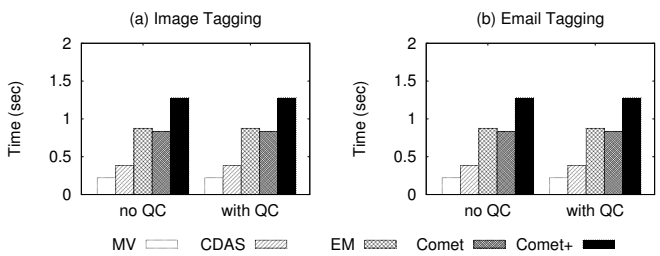

Figure 7: Truth Inference (Efficiency Comparisons).

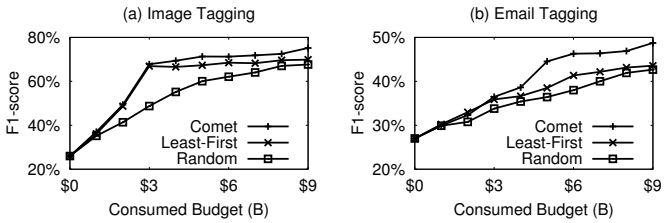

Figure 8: Task Assignment (Effectiveness Comparisons).

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. Especially, for all datasets, we have $\tau(c) \leq 10^{-3}$ as $c \geq 20$. 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 $c \geq 20$. Thus it is quick to converge (≤ 20 iterations).

Label Correlations. In Figure 5(d), we observe the effect of varying label correlation parameter α in Equation 7. $\alpha \in [0, 1]$ measures the impact of $S_{i,j}$ (itself) and related labels. 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 $\alpha = 0.7$.

Comparisons

Effectiveness. 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 8. We can observe that (1) our proposed approaches perform much better on datasets with no QC, where Comet and Comet+ lead other competitors for more than 22%. As can be seen from Figures 6(a)(c), 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. However, Comet 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, Comet (Comet+) 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, especially for image tagging (can be seen in Figure 4(b)), we can still outperform 2.08%. (3) By considering label correlations, Comet+ improves Comet 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. Based on the ground truth, we further analyze image tagging, finding that most of the strong label correlations are caught by the results of Comet (computed purely based on workers' answers), and there are only < 20 pairs that can be further benefited by considering $\mathcal{M}(\cdot)$, resulting in Comet+'s limited improvement. However, email tagging tasks are more difficult for workers and most of the label correlations are not caught by Comet (e.g., the high value of $\mathcal{M}(\cdot)$ on 'California energy crisis' and 'company business&strategy'). Similarly, we analyze the results of Comet on email tagging, finding that there are > 200 pairs that can be further benefited by considering $\mathcal{M}(\cdot)$, thus Comet+ can improve a lot compared with Comet.

Efficiency. We compare the execution time in Figure 7. Given that truth inference can be addressed off-line, we can observe that all methods are efficient. To be specific, they all can be finished within 1.5s. MV and CDAS are more efficient as they do not need many computations; EM and Comet adopt iterative approaches which require more computations; Comet+ considers label correlations in the iterative approach, which is the least efficient. We will evaluate the scalability of Comet+ on simulated data (Section 7.5).

Table 8: Truth Inference (Effectiveness: F1-score).

Method	*Image Tagging*		*Email Tagging*	
	no QC	*with QC*	*no QC*	*with QC*
MV	29.28%	76.26%	2.98%	44.36%
CDAS	28.79%	76.26%	3.30%	22.42%
EM	29.18%	76.28%	11.10%	44.32%
Comet	53.25%	78.36%	33.79%	48.75%
Comet+	53.40%	78.36%	45.95%	54.35%

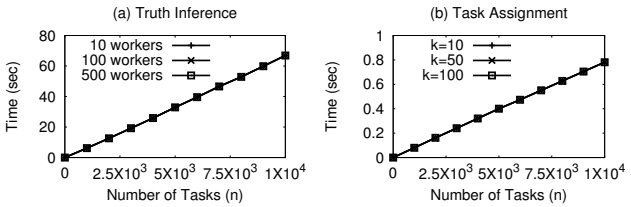

Figure 9: Scalability on Simulated Data.

7.4 Task Assignment

We evaluate the effectiveness and efficiency of Comet's task assignment (Section 6). As existing works [11,27,47] that study task assignment focuses on single-label tasks, and they are not straightforward to extend to multi-label tasks. Thus we compare with two baseline assignment strategies for multi-label tasks.

Random: it randomly selects k tasks;

Least-First: it selects k tasks that have been assigned least times.

We use two datasets (image tagging and email tagging) and perform experiments on AMT [1] with quality control as specified in Section 7.1. 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. When a worker comes, we select $k = 3$ tasks and batch them in a HIT for the coming worker. To control the effect of workers, for one dataset, we perform 3 experiments on AMT in parallel, where each one uses a specific assignment strategy.

Effectiveness. We compare the effectiveness of different assignment strategies in Figure 8. For a fair comparison, we use Comet's truth inference method (Algorithm 1) 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 Comet outperforms both Least-First and Random. We can also observe in Figures 8(a)(b) that Least-First and Comet 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 while leaving a few tasks not selected. As B increases, Comet can measure the reduction in uncertainty when selecting tasks, thus gradually outperforming other competitors. In Table 9, we give an exact comparison of F1-score as $B = \$9$ for different assignment strategies on two datasets. It shows that Comet leads more than 5% compared with the best of other competitors, and the quality comparison result can be generally summarized as follows: Comet>Least-First>Random.

Efficiency. In Table 9, we also 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. Comet is the least efficient, as it computes the uncertainty reduction for each task, and chooses k tasks with the highest reduction in uncertainty; Least-First needs to decide the tasks that are answered with the least times, which is more expensive than Random. We evaluate the scalability of Comet on simulated data in next Section.

Table 9: Task Assignment (Effectiveness and Efficiency).

Method	*Image Tagging*		*Email Tagging*	
	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
Comet	75.19%	0.025s	48.76%	0.023s

7.5 Scalability on Simulated Data

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

Truth Inference. In Figure 9(a), we evaluate the scalability of Comet's truth inference method (Algorithm 1). 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 $n \in [0, 10^4]$, $|\mathcal{W}| \in \{10, 100, 500\}$ and run truth inference on randomly generated workers' answers. It can be seen from Figure 9(a) that (1) for a fixed $|\mathcal{W}|$, the time linearly increases with n; when n is big enough (e.g., 10^4), the time is within 70s, which is efficient (as the truth inference can be done off-line). (2) For a fixed n, the time does not change with varying $|\mathcal{W}|$. It may contradict to the complexity, i.e., $\mathcal{O}(c \cdot |\mathcal{W}| \cdot \sum_{i=1}^{n} |L_i|)$. However, it is an upper bound, and we make further analysis to explain it: for a fixed budget, a larger $|\mathcal{W}|$ means the participation of more workers, thus the average number of tasks a worker has answered is smaller, leading to the fact the computation for each worker is smaller. This may explain why the time is invariant with $|\mathcal{W}|$ in practice.

Task Assignment. In Figure 9(b), we evaluate the scalability of Comet's assignment strategy (Section 6). 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 $n \in [0, 10^4]$, $k \in \{10, 50, 100\}$, and run Comet to assign k tasks for worker w, and record the time. It can be observed in Figure 9(b) that (1) for a fixed k, the time linearly increases with n, corresponding to the complexity $\mathcal{O}(\sum_{i=1}^{n} |L_i|)$; (2) for a fixed n, the time does not change if k varies, because after we compute the uncertainty reduction of each object, we can use the PICK algorithm [10] to select top-k objects, which is invariant with k.

8. RELATED WORKS

Crowdsourced Data Management. With the development in crowdsourcing [25], several platforms are built (e.g., AMT [1], CrowdFlower [3], ChinaCrowd [2]), enabling crowd workers to perform tasks. To facilitate query processing, crowdsourced databases (e.g., CrowdDB [20], Deco [33], Qurk [29]) are built on top of crowdsourcing platforms to support crowdsourced queries, e.g., join [37–39], max [21,36], selection [32,34], top-k [15], sort [28], etc.

Multi-Label Tasks. In machine-learning field, multi-label tasks have been widely studied [9,42,45] and applied to many applications, e.g., text categorization [30], bioinformatics [7]. 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 [9,42,45] can better facilitate our truth inference. In crowdsourcing, multi-label tasks are addressed [32,47] based on transforming each task to many *independent* single-label tasks, which will incur more latency and budget [17]. Our method is different from them in that we use multi-

label tasks, and our worker model can better capture a worker's quality. We also consider label correlations and online task assignment. Note that there are also crowdsourcing works [12,14,17, 22] on multi-label tasks, but with different objectives, e.g., [12,14] study how to build a taxonomy tree for an item, and [22] 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) [13,31] do not model a worker and treat each worker equally; (2) [27,43,46] model a worker using Accuracy; (3) [23,32,47] 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.

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 [13,31], which treats each worker equally; (2) [27,43,46] leverage Bayesian Voting to infer truth, which model Accuracy for each worker; (3) [23, 47] use EM algorithm 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 more challenging. Note that there are some data integration works [18, 41] that also address the truth inference problem, but their settings are quite different from ours as (1) they do not model workers, and (2) they cannot actively select tasks and assign to online workers.

Task Assignment. The task assignment problem has been studied in existing crowdsourcing works, mostly on single-label tasks [11, 27,47]. Typically they measure the uncertainty of each task and assign the most uncertain tasks to the coming worker. We are different in that we address task assignment on multi-label tasks, where an answer for a task contains votes of multiple labels from a worker. Intuitively, we measure the uncertainty reduction (rather than the uncertainty) of a task by considering if it is answered by the coming worker and select tasks with the highest uncertainty reduction.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we examine two fundamental problems about multi-label tasks: *Truth Inference Problem* and *Task Assignment Problem*. In the first problem, we model a worker as Precision and Recall, and develop an iterative approach that captures the relations between truth and workers' qualities. We further incorporate label correlations in computing the truth. In the second problem, when a worker comes, we select the tasks whose uncertainty can be reduced the most for the worker. We perform experiments both on real-world datasets and simulated data, verifying that our proposed approaches are robust, outperforming existing methods with various workers, and also scalable to large datasets. In future work, we will study different User Interfaces (UIs) of multi-label tasks, and the relations among the number of labels, quality, and latency, etc.

10. REFERENCES

[1] Amazon mechanical turk. https://www.mturk.com/.

[2] Chinacrowd. http://www.chinacrowds.com.

[3] Crowdflower. http://www.crowdflower.com.

[4] Flickr. https://www.flickr.com/.

[5] Sigmoid function. https://en.wikipedia.org/wiki/Sigmoid_function.

[6] Technical report. http://i.cs.hku.hk/~ydzheng2/comet_full.pdf.

[7] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[8] Bayes' theorem. https://en.wikipedia.org/wiki/Bayes%27_theorem.

[9] W. Bi and J. T. Kwok. Multilabel classification with label correlations and missing labels. In *AAAI*, 2014.

[10] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Time bounds for selection. *JCSS*, 1973.

[11] 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*.

[12] J. Bragg, D. S. Weld, et al. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*, 2013.

[13] 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.

[14] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay. Cascade: Crowdsourcing taxonomy creation. In *CHI*, 2013.

[15] 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.

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

[17] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei. Scalable multi-label annotation. In *SIGCHI*, 2014.

[18] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2009.

[19] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV 2002*, pages 97–112. Springer, 2002.

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

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

[22] H. Hu, Y. Zheng, Z. Bao, G. Li, J. Feng, and R. Cheng. Crowdsourced poi labelling: Location-aware result inference and task assignment. In *ICDE*, 2016.

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

[24] B. Klimt and Y. Yang. Introducing the enron corpus. In *Proceedinds of the 1st Conference on Email and Anti-Spam*, 2004.

[25] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *TKDE*, 2015.

[26] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[27] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang. CDAS: A crowdsourcing data analytics system. *PVLDB*, 2012.

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

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

[30] A. K. McCallum. Multi-label text classification with a mixture model trained by em. In *AAAI Workshop on Text Learning*. Citeseer, 1999.

[31] L. Mo, R. Cheng, B. Kao, X. S. Yang, C. Ren, S. Lei, D. W. Cheung, and E. Lo. Optimizing plurality for human intelligence tasks. In *CIKM*, 2013.

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

[33] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, 2012.

[34] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and A. Y. Halevy. Crowd-powered find algorithms. In *ICDE*, 2014.

[35] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, Jan. 2001.

[36] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis. Max algorithms in crowdsourcing environments. In *WWW*, 2012.

[37] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. CrowdER: crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[38] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng. Leveraging transitive relations for crowdsourced joins. In *SIGMOD*, 2013.

[39] S. E. Whang, P. Lofgren, and H. Garcia-Molina. Question selection for crowd entity resolution. *PVLDB*, 6(6):349–360, 2013.

[40] word2vec. https://code.google.com/p/word2vec/.

[41] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *TKDE*, 20(6):796–808, 2008.

[42] Y. Yu, W. Pedrycz, and D. Miao. Multi-label classification by exploiting label correlations. *Expert Systems with Applications*, 2014.

[43] C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *PVLDB*, 2013.

[44] C. J. Zhang, Y. Tong, and L. Chen. Where to: Crowd-aided path selection. *PVLDB*, 7(14):2005–2016, 2014.

[45] M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *KDD*, pages 999–1008. ACM, 2010.

[46] Y. Zheng, R. Cheng, S. Maniu, and L. Mo. On optimality of jury selection in crowdsourcing. In *EDBT*, pages 193–204, 2015.

[47] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *SIGMOD*, 2015.

11. APPENDIX

In the appendix, we first show the incremental computation of truth inference (Section 11.1), and then present the detailed proofs for Theorems 1- 5 (Sections 11.2-11.6).

11.1 Incremental Computation

As Algorithm 1 takes an iterative approach, we now discuss how to incrementally update the two sets of parameters (i.e., all truth and workers' models). The reason why it comes to our interest is that Algorithm 1 solves 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 that instantly updates previously stored parameters when a new answer arrives. The basic idea is that upon receiving a worker's new answer, we only choose a small subset of *related* parameters to update, e.g., the truth of voted pairs, and all workers' models who have ever answered the voted pairs. The challenge is that (1) when a worker comes, which subset of parameters are to update; (2) what parameters should be stored for each pair and each worker, such that the update can be facilitated. Note that the designed algorithm may not achieve as high quality as the iterative approach, the benefit of this algorithm is its efficiency and applicability to a high data velocity.

To address the challenge, we develop the incremental algorithm in Algorithm 2. To be precise,
(1) for a worker $w \in \mathcal{W}$, we store 4 parameters: ETP_w, EFP_w, EFN_w, and ETN_w, representing the expectations of TP, FP, FN and TN, 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, EFP_w, and ETN_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.

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 $1 \le j \le |L_i|$, $(\text{ETP}_w, \text{EFP}_w, \text{EFN}_w, \text{ETN}_w)$ for $w \in \mathcal{W}$
Output: V, $(m^{\text{Y}}_{i,j}, m^{\text{N}}_{i,j})$ for $1 \le i \le n$ and $1 \le j \le |L_i|$, $(\text{ETP}_w, \text{EFP}_w, \text{EFN}_w, \text{ETN}_w)$ for $w \in \mathcal{W}$

1 $\text{tempT} = m^{\text{Y}}_{i,j}$, $\text{tempN} = m^{\text{N}}_{i,j}$;
2 // (1) Inferring the Truth
3 **if** $v = Y$ **then**
4 $\quad m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot \frac{\text{ETP}_w}{\text{ETP}_w+\text{EFP}_w}$;
5 $\quad m^{\text{N}}_{i,j} = m^{\text{N}}_{i,j} \cdot \frac{\text{EFP}_w}{\text{ETP}_w+\text{EFN}_w}$;
6 **else**
7 $\quad m^{\text{Y}}_{i,j} = m^{\text{Y}}_{i,j} \cdot (1 - \frac{\text{ETP}_w}{\text{ETP}_w+\text{EFP}_w})$;
8 $\quad m^{\text{N}}_{i,j} = m^{\text{N}}_{i,j} \cdot (1 - \frac{\text{EFP}_w}{\text{ETP}_w+\text{EFN}_w})$;
9 // (2) Estimating Workers' Models
10 // (2.1) Update the parameters for worker w
11 **if** $v = Y$ **then**
12 $\quad \text{ETP}_w = \text{ETP}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
13 $\quad \text{EFP}_w = \text{EFP}_w + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
14 **else**
15 $\quad \text{EFN}_w = \text{EFN}_w + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
16 $\quad \text{ETN}_w = \text{ETN}_w + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
17 // (2.2) Update the parameters for workers who have voted the pair $(o_i, \ell_{i,j})$ before
18 **for** $(w', v') \in V_{i,j}$ **do**
19 $\quad$ **if** $v' = Y$ **then**
20 $\quad\quad \text{ETP}_{w'} = \text{ETP}_{w'} - \frac{\text{tempT}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
21 $\quad\quad \text{EFP}_{w'} = \text{EFP}_{w'} - \frac{\text{tempN}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
22 $\quad$ **else**
23 $\quad\quad \text{EFN}_{w'} = \text{EFN}_{w'} - \frac{\text{tempT}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{Y}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
24 $\quad\quad \text{ETN}_{w'} = \text{ETN}_{w'} - \frac{\text{tempN}}{\text{tempT}+\text{tempN}} + \frac{m^{\text{N}}_{i,j}}{m^{\text{Y}}_{i,j}+m^{\text{N}}_{i,j}}$;
25 $V_{i,j} = V_{i,j} \cup \{(w, v)\}$;
26 **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, $ETN_w)$ *for* $w \in \mathcal{W}$;

11.2 Proof for Theorem 1

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

PROOF. To prove it, similar to the main contents, by assuming uniform priors (i.e., $\Pr(t_{i,j} = \mathrm{Y}) = \Pr(t_{i,j} = \mathrm{N}) = 0.5$), we have

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

Based on Equation 2, the above formula is indeed

$$p_w = \frac{r_w}{r_w + \Pr(v^w_{i,j} = \mathrm{Y} \mid t_{i,j} = \mathrm{N})},$$

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

□

11.3 Proof for Theorem 2

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

PROOF. Based on the Bayes' Theorem, considering $t_{i,j} \in \{\mathrm{Y}, \mathrm{N}\}$ we can derive that

$$\begin{aligned}\Pr(v^w_{i,j} = \mathrm{Y} \mid V) = &\Pr(v^w_{i,j} = \mathrm{Y} \mid t_{i,j} = \mathrm{Y}, V) \cdot \Pr(t_{i,j} = \mathrm{Y} \mid V) \\ &+ \Pr(v^w_{i,j} = \mathrm{Y} \mid t_{i,j} = \mathrm{N}, V) \cdot \Pr(t_{i,j} = \mathrm{N} \mid V).\end{aligned}$$

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

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

□

11.4 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 $q^{\mathrm{Y}}_{i,j} = q_{i,j}/[\, q_{i,j} + (1 - q_{i,j}) \cdot \frac{1 - p_w}{p_w} \,]$; otherwise, if the worker's new vote is N, then $q_{i,j}$ is updated to $q^{\mathrm{N}}_{i,j} = q_{i,j}/[\, q_{i,j} + (1 - q_{i,j}) \cdot \frac{1 - r_w/p_w + r_w}{1 - r_w} \,]$.*

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

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

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

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

By dividing $A \cdot [\, \Pr(V_{i,j} \mid t_{i,j} = \mathrm{Y}) + \Pr(V_{i,j} \mid t_{i,j} = \mathrm{N}) \,]$ 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 $q^{\mathrm{Y}}_{i,j} = q_{i,j}/[\, q_{i,j} + (1 - q_{i,j}) \cdot \frac{1 - p_w}{p_w} \,]$; otherwise, if the worker's new vote is N, then $q_{i,j}$ is updated as $q^{\mathrm{N}}_{i,j} = q_{i,j}/[\, q_{i,j} + (1 - q_{i,j}) \cdot \frac{1 - r_w/p_w + r_w}{1 - r_w} \,]$.

□

11.5 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.,*

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

PROOF. Given the following formula

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

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 $\sigma_{|L_i|} = \mathrm{Y}$ (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 $\sigma_{|L_i|} = \mathrm{N}$ (worker w votes N for $(o_i, \ell_{i,|L_i|})$), i.e.,

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

With a careful deduction, we can derive that

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

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

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

□

11.6 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.,

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

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.

□