Cleaning Crowdsourced Labels Using Oracles For Supervised Learning

Mohamad Dolatshah Mathew Teoh Jiannan Wang Jian Pei

Simon Fraser University
{mteoh, mdolatsh, jnwang, jpei}@sfu.ca

ABSTRACT

Nowadays, crowdsourcing is being widely used to collect training data for supervised learning. However, crowdsourced labels are often noisy, and there is a performance gap between learning with noisy labels and learning with true labels. In this paper, we consider how to apply oracle-based label cleaning to reduce the gap. We propose TARS, a label-cleaning advisor that can provide two pieces of valuable advice for data scientists when they need to train or/and test a model using noisy labels. Firstly, in the model testing stage, given a test dataset with noisy labels, and a classification model, TARS can use the test data to estimate how well the model will perform w.r.t. true labels. Secondly, in the model training stage, given a training dataset with noisy labels, and a supervised-learning algorithm, TARS can determine which label should be sent to an oracle to clean such that the model can be improved the most. For the first advice, we propose an effective estimation technique, and study how to compute confidence intervals to bound its estimation error. For the second advice, we propose a new cleaning strategy along with two optimization techniques, and illustrate that it is superior to the existing cleaning strategies. We evaluate the effectiveness of TARS on both simulated and real-world datasets. The results show that (1) TARS can use noisy test data to accurately estimate a model's true performance for various evaluation metrics (e.g., Accuracy, F-Score); and (2) TARS can improve the model accuracy by a larger margin than the existing cleaning strategies, for the same cleaning budget.

1 INTRODUCTION

Supervised learning, which can solve a large variety of real-world problems, such as spam detection, entity resolution, sentiment analysis, and image classification, has recently attracted significant attention in both industry and academia. Its basic idea is to learn a function from a collection of ⟨instance, label⟩ pairs such that the function can predict the labels of unseen instances. The resulting function is also known as a *model* or a *classifier*. Typically, the collection of labeled pairs is split into two parts: training data and test data, where the training data is for model training and the test data is for model evaluation.

Despite the great success that supervised learning has achieved, a fundamental limitation of using it in practice is the high cost of data labeling. Crowdsourcing is a promising way to solve this problem. Crowdsourcing platforms such as Amazon Mechinical Turk have hundreds of thousands of crowd workers available. These workers can be used to label data at low cost and fast speed. But at the same time, crowd workers are not accurate. They often provide *noisy labels* with certain probabilities to be wrong. Although this issue can be mitigated by assigning an instance to multiple workers and then infer the instance's true label using a truth-inference algorithm like Majority Vote, the state-of-the-art truth-inference algorithms are still far from perfect [39].

Having noisy labels in training data will negatively affect the performance of a supervised-learning algorithm because the algorithm tries to predict noisy labels rather than true labels. In the Machine Learning community, there has already been some work that studies how to clean noisy labels to solve this problem [3]. However, the noisy labels are cleaned by heuristic algorithms, which have no guarantee on cleaning accuracy and may even mess up a lot of correct labels [12].

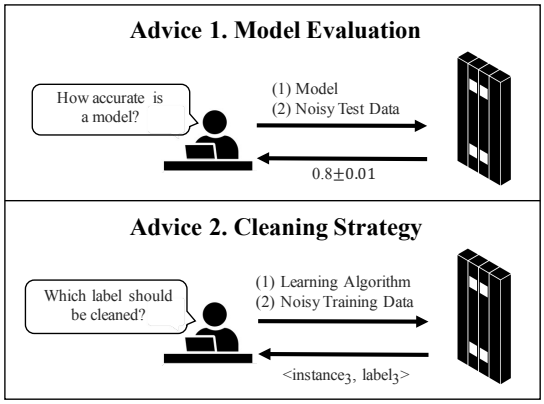


Figure 1: TARS can provide two pieces of valuable advice for data scientists when they need to train or/and test a model using noisy labels

Unlike the existing work, our work draws some inspiration from the recent progress in the data cleaning community [1, 11, 16], and focuses on a *different data-cleaning scenario*. We consider that there exists an oracle who can be queried to clean noisy labels. Each query is to ask the oracle to verify whether a training example, ⟨instance, label⟩, is correctly labeled or not. If not, replace its label with the true label. This scenario more often than not holds in reality. Imagine a data scientist needs to train a good model on noisy data. In this situation, she can ask internal experts from her company to serve as oracles to clean the noisy labels.

It is worth noting that the goal of this paper is *not* to develop yet another supervised learning algorithm for noisy labels. Instead, we aim to develop a *label cleaning advisor*, named as TARS[1], that can advise a data scientist on how to make the best use of oracle-based cleaning for supervised learning. As shown in Figure 1, suppose a data scientist has already trained a model on a noisy dataset. The first piece of advice she may ask for is how accurate the model is. If the current model is already good enough, there is no need to further spend effort on label cleaning. However, if she finds that the model does not meet her need, the next question she may ask is which label should be cleaned such that the model can be improved the most. TARS can provide the two pieces of advice:

- *Advice 1. Model Evaluation.* Given a model, and a test dataset labeled by the crowd, TARS can tell a data scientist how well

[1] TARS is named after an intelligent robot in *Interstellar* who can provide insightful advice for human beings.

(a) Ground-truth Labels

Instance	True Label
x_1	+1
x_2	+1
x_3	-1
x_4	-1
x_5	-1

(b) Crowdsourced (noisy) labels

Instance	Noisy Label
x_1	$(w_1,+1)$
x_2	$(w_1,+1)$
x_3	$(w_1,+1)$
x_4	$(w_1,-1)$
x_5	$(w_2,-1)$

(c) Model's Prediction

Instance	Predicted Label
x_1	+1
x_2	+1
x_3	-1
x_4	-1
x_5	+1

Figure 2: Datasets with ground-truth labels, crowdsourced labels, and a model's predicted labels.

the model will perform w.r.t. *true labels* (rather than w.r.t. noisy labels).

- *Advice 2. Cleaning Strategy.* Given a learning algorithm, and a training dataset labeled by the crowd, TARS can tell a data scientist which label in the training dataset should be cleaned such that the new model, re-trained on the cleaned training dataset using the learning algorithm, has the best performance.

There are some straightforward solutions to the above two problems. In the following, we will use the simple example in Figure 2 to illustrate the limitations of these solutions, and then demonstrate the contributions made in this paper to overcome the limitations.

Let us first consider Advice 1. Figure 2(b) shows a test dataset labeled by two crowd workers w_1 and w_2. For simplicity, we assume that the workers w_1 and w_2 have the same noise rate = 0.2, which means that each of them has a probability of 0.2 to give a different label from the true label. We apply a given model to the test dataset and obtain the predicted label of each instance (see Table (c)). Since true labels are unknown, the model's true accuracy cannot be directly derived. One naive approach is to treat noisy labels as true labels and then compute the accuracy based on the noisy labels. However, this approach is biased because it ignores the workers' noise rates. In this example, the accuracies computed based on the noisy labels and the true labels are $\frac{3}{5}$ and $\frac{4}{5}$, respectively, and the difference is $\frac{1}{5}$.

To overcome the limitation, we present a new estimator that estimates the model's true performance (e.g., accuracy, precision, recall, or F-score) by considering not only noisy labels but also noise rates. We prove that our estimator is unbiased (i.e., in expectation the estimator's estimated value is equal to the true value). We further study how to compute a confidence interval for the estimator in order to bound its estimation error (i.e., bound the difference between the estimated value and the true value). This turns to be a challenging problem because the estimator's error comes from two sources: sampling and labeling. We theoretically analyze how each source contributes to the overall estimation error, and show that the contribution of the first (second) source is controlled by test data size N (a noise rate β). Interestingly, they will not be affected by each other: (i) as N increases, the overall estimation error will decrease at a rate of $O(\frac{1}{\sqrt{N}})$, regardless of what β is, and (ii) as β decreases, the overall estimation error will decrease at a rate of $O\left(\frac{1}{|0.5-\beta|}\right)$, regardless of what N is. In other words, the estimation error can be decreased by either increasing test data size or improving worker quality; the above theoretical results show the trade-off of each choice.

Next, let us consider Advice 2. Suppose Figure 2(b) represents a training dataset, where w_1's noise rate is 0.4 and w_2's noise rate is 0.01. There are five noisy labels in the dataset, and TARS needs to decide which one should be sent to an oracle to clean. Please note that this problem is different from active learning [29] because active learning assumes that data is unlabeled but here data has been labeled by crowd workers. We need to incorporate label noise into our cleaning strategy otherwise an oracle may clean many instances that have already been correctly labeled. For example, consider the data in Figure 2(b), the first four instances have the label noise of 0.4 and the label noise of the last instance is only 0.01. A good cleaning strategy should try to avoid sending the last instance to an oracle because it has a probability of 0.99 to be correct. (In addition to active learning, there are some other cleaning strategies proposed in the literature. Please refer to Section 5.1 for a more detailed discussion.)

To this end, we propose a new cleaning strategy, called expected model improvement (EMI). EMI estimates the expected model improvement of cleaning each noisy label and then selects the noisy label with the largest estimated value to clean. We illustrate the limitations of the existing cleaning strategies and explain why EMI can overcome the limitations. While the idea of EMI sounds promising in theory, we need to address some practical issues. The first issue is which data should be used to estimate the expected model improvement. If we choose the data improperly, EMI may end up training a model that performs well on the training data but not on the test data. The second issue is how to break the tie when two noisy labels have the same expected model improvement. We propose optimization techniques to address the issues and demonstrate their effectiveness experimentally.

Note that Figure 2 only shows a simplified version of our problem. In the paper, we study a more general version of the problem, where an instance can be labeled by multiple workers, a confusion matrix is used to model worker quality, and various metrics such as accuracy, precision, recall, and F-score can be chosen for model evaluation. In summary, our paper makes the following contributions:

- To the best of our knowledge, we are the first to study how to use an oracle to clean crowdsourced labels for supervised learning. We identify two challenging problems (model evaluation and cleaning strategy) in this study and present the formal problem definitions.
- We propose an estimator that can estimate a model's true accuracy based on noisy data. We prove that the estimator is unbiased and we compute a confidence interval to bound its estimation error. We also discuss how to extend our solution to other evaluation metrics such as precision, recall, F-score.
- We develop a new cleaning strategy, called EMI, that can effectively decide which label should be cleaned. We explain why EMI is superior to the existing cleaning strategies to solve our problem. We further improve the effectiveness of EMI by developing two optimization techniques.
- We evaluate TARS on both simulated and real-world datasets. The results show that (1) TARS can use noisy labels to accurately estimate how well a model will perform w.r.t. true labels; and (2) TARS can improve the model accuracy by a larger margin than the existing cleaning strategies, for the same cleaning budget.

The remainder of this paper is organized as follows. Section 2 formally defines the model evaluation and cleaning strategy problems. Since each instance may be labeled by multiple workers, we introduce how label consolidation works in Section 3. After that, we discuss how to solve the model evaluation problem in Section 4 and the cleaning strategy problem in Section 5. Experimental results are presented in Section 6, followed by related work (Section 7) and conclusion (Section 8).

2 BACKGROUND AND PROBLEM FORMALIZATION

We first provide some background knowledge in Section 2.1, and then formally define our problems in Sections 2.2 and 2.3.

2.1 Background

Learning With True Labels. Let $\mathbb{G}$ be the joint distribution on $(x, y) \in X \times Y$, where x represents an input instance (typically a vector) and $y \in \{-1, +1\}$ represents the true label. Denote a sample drawn i.i.d. from $\mathbb{G}$ as $S = \{(x_i, y_i)\}_{i=1}^{N}$. A supervised learning algorithm aims to train a *predictive model (i.e., a classifier)* f based on S and then make predictions over the unseen instances in $\mathbb{G}$, where f can be thought of as a decision function that takes an instance x as input and outputs a real value $t = f(x)$. The instance will be classified as +1 if $t > 0$, and -1 otherwise[2].

Crowdsourced Data. Obtaining ground truth labels can be expensive, but labeling instances with imprecise crowd workers can be cheap. Let $\mathcal{W} = \{w_j\}_{j=1}^{K}$ denote a set of workers. We assume that each instance x_i is labeled by a subset of k_i workers; let $\mathcal{L}^i = \{(w_j, l_{i,j}) \mid w_j \text{ labels } x_i\}$ denote the corresponding labels, where $l_{i,j} \in \{-1, +1\}$ represents the label given by worker w_j. Let $C = \{(x_i, \mathcal{L}^i)\}_{i=1}^{N}$, which we call the *crowdsourced data*.

Worker Model. Since crowd workers are not perfect, existing crowdsourcing work typically uses *confusion matrix* to model worker quality. The confusion matrix of each worker w_j is a 2×2 matrix,

$$q^{(j)} = \begin{bmatrix} q^{(j)}_{-1,-1}, & q^{(j)}_{-1,+1} \\ q^{(j)}_{+1,-1}, & q^{(j)}_{+1,+1} \end{bmatrix},$$

where each row represents a true label, each column represents a worker's provided label, and $q^{(j)}_{y,l}$ ($y \in \{\pm 1\}, l \in \{\pm 1\}$) means that given an instance with true label y, worker w_j provides label l with probability of $q^{(j)}_{y,l}$. For example, the following illustrates the confusion matrices of a perfect worker w_1 and an imperfect worker w_2 (with the noise rates of 0.2 given $y = -1$ and 0.3 given $y = +1$):

$$q^{(1)} = \begin{bmatrix} 1, & 0 \\ 0, & 1 \end{bmatrix}, \quad q^{(2)} = \begin{bmatrix} 0.8, & 0.2 \\ 0.3, & 0.7 \end{bmatrix}.$$

Prior Probability. Each cell in a confusion matrix is a conditional probability, $q^{(j)}_{y,l} = P(L = l \mid Y = y)$. To compute the joint probability distribution $P(L, Y)$, we also need to know a *prior probability* (or simply called the prior), denoted by $P(Y)$. Intuitively, the prior is the probability of an instance having a label of +1 or -1.

Computing Confusion Matrices and Prior. An important problem is how to compute workers' confusion matrices $q^{(j)}$ (for all $j \in [1, K]$) and the prior $P(Y)$ in practice. This problem has been extensively studied in the crowdsourcing literature [18]. One simple idea is to manually label a small sample of instances upfront, and then mix these instances with other unlabeled instances and ask workers to label them all. Since workers do not know which instances have been pre-labeled, these pre-labeled instances can be used to compute workers' confusion matrices as well as the prior. Another common idea is to leverage label redundancy. To improve quality, each instance is often labeled by multiple workers. If a worker often provides inconsistent labels with the majority of other workers, then the worker is very likely to be a low-quality worker. Based on this idea, existing work treats confusion matrices and the prior as unknown parameters and adopts an EM algorithm [8] to iteratively estimate their values.

[2]For ease of presentation, we only consider binary classification in this paper, but the proposed approaches can be extended to multiclass classification by thinking of the problem of multiclass classification as multiple binary classification problems.

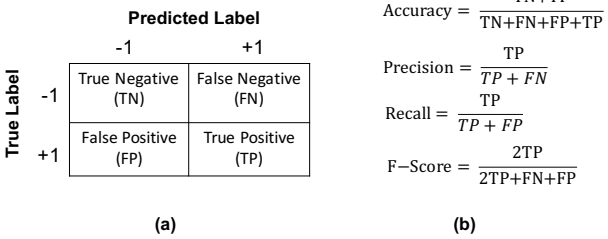


Figure 3: An illustration of (a) model's confusion matrix and (b) four evaluation metrics derived from the confusion matrix.

Assumption. In this paper, we treat computing workers' confusion matrices and the prior as an orthogonal problem, and we assume that they are given as input of our problems.

Learning With Noisy Labels. Let $\mathcal{A}$ denote a supervised learning algorithm that learns a model f on crowdsourced data C. This problem has been well studied in the Machine Learning community [12, 26]. Their basic ideas are either to develop a robust learning algorithm to tolerate label noise or to adopt automatic cleaning algorithms to filter/correct noisy labels. In this paper, we treat $\mathcal{A}$ as a black box, which takes crowdsourced data C as input and outputs a classifier

2.2 Advice 1: How Good is a Model?

The first piece of advice from TARS is focused on the *model testing* stage. It considers the situation when a user has already trained a model and wants to evaluate the model's performance using crowdsourced data.

Evaluation Metrics. To evaluate a model's performance, people often compute a confusion matrix for the model and then derive different types of evaluation metrics from the matrix. Figure 3(a) illustrates a model's confusion matrix. We can see that it is similar to a worker's confusion matrix, where each row also represents a true label, but the difference is that each column represents a model's predicted label rather than a worker's provided label. The matrix has four cells:

- *True Positive (TP)*: the number of positive instances that are correctly predicted by a model;
- *False Positive (FP)*: the number of positive instances that are falsely predicted by a model;
- *True Negative (TN)*: the number of negative instances that are correctly predicted by a model;
- *False Negative (FN)*: the number of negative instances that are falsely predicted by a model.

TARS aims to estimate the value of each cell. In this way, any evaluation metric computed based on these cells can be derived in a straightforward manner. Figure 3(b) shows the definitions of four representative evaluation metrics.

Let eval denote a user-specified evaluation metric. The model's *true performance* is denoted by $\text{eval}(\mathbb{G}, f)$. For example, suppose eval is accuracy, then $\text{eval}(\mathbb{G}, f)$ computes the f's accuracy on $\mathbb{G}$. Since we do not have access to $\mathbb{G}$ but only crowdsourced data C, TARS aims to use C to estimate $\text{eval}(\mathbb{G}, f)$. Let $\widetilde{\text{eval}}(C, f)$ denote an estimation of $\text{eval}(\mathbb{G}, f)$. We say $\widetilde{\text{eval}}(C, f)$ to be unbiased if the expected value of the estimation is equal to the true value, i.e., $\mathbb{E}[\widetilde{\text{eval}}(C, f)] = \text{eval}(\mathbb{G}, f)$.

In addition, TARS computes a *confidence interval* to bound the estimation error of $\widetilde{\text{eval}}(C, f)$. Suppose the estimated value is $\widetilde{\text{eval}}(C, f) = 0.8$. Given a confidence level (e.g., 95%), a confidence interval (e.g., 0.8 ± 0.01) indicates that the difference between the estimated value

and the true value is within ±0.01 with 95% probability. The wider the confidence interval, the larger the estimation error.

PROBLEM 1 (MODEL EVALUATION). *Given crowdsourced data C, a model f, and an evaluation metric* eval, TARS *aims to determine (1) an unbiased estimator, $\widetilde{eval}(C, f)$, of the model's true performance, and (2) a confidence interval, $[\widetilde{eval}(C, f) - \epsilon_1, \widetilde{eval}(C, f) + \epsilon_2]$, at a given confidence level.*

2.3 Advice 2: Which Label Should be Cleaned?

The second piece of advice that TARS can provide is focused on the *model training* stage. It considers the situation when a user trains a model using noisy data, but the model is not good enough. The user wants to know which instance-label pair should be sent to an *oracle* to clean such that the model's true performance can be improved the most.

Oracle Labeller. Cleaning, or ground truth labeling, can be thought of as querying a perfect worker w_Ω called an *oracle*. It follows that the noise rates for w_Ω are $P(L = -1 \mid Y = +1) = P(L = +1 \mid Y = -1) = 0$. We assume that queries are expensive, and thus calls to the oracle are constrained by a budget.

Cleaning Data. Suppose we query an oracle to clean an instance x_i and obtain ground truth label y_i. One could update crowdsourced data C with this new knowledge by replacing $\mathcal{L}^i$ with $\{(w_\Omega, y_i)\}$. We say that we are *cleaning* instance x_i, because we are substituting a set of imprecise labels with the ground truth label provided by the oracle.

Cleaning Strategy. Recall that a learning algorithm $\mathcal{A}$ takes crowdsourced data C as input and outputs a model f. By cleaning the labels in C, we hope $\mathcal{A}$ can produce a better model. It is expensive to query an oracle to get ground truth labels, thus the goal is to strategically choose x_i to clean. The best choice of x_i would result in a dataset which leads to a model more accurate than cleaning any other label. Based on the notation above, we have a formal definition of the problem below:

PROBLEM 2 (CLEANING STRATEGY). *Given crowdsourced data C, a learning algorithm $\mathcal{A}$, and an evaluation metric* eval, *let f_i be the model resulting from cleaning instance x_i, then training on C with $\mathcal{A}$.* TARS *aims to determine which instance should be cleaned such that the model's true performance can be improved the most:*

$$i^* = \underset{i=1,\ldots,N}{\operatorname{argmax}}\, eval(\mathbb{G}, f_i).$$

3 PRELIMINARY: LABEL CONSOLIDATION

Given crowdsourced data, if there are instances in the data having multiple worker labels, TARS will first consolidate these label into a single label. In other words, after this process, each instance in the crowdsourced data will have a single consolidated label along with a confusion matrix that quantifies the uncertainty of the consolidated label. In this section, we will start by showing why there is a need for this process, and then present how to get the consolidated label as well as the consolidated confusion matrix.

3.1 The Need For Label Consolidation

Both Problems 1 and 2 require estimating the performance of some model which is defined based on the true labels in G. Rather than true labels, we only have worker labels, which are noisy. The trick to bridging the gap between worker labels and true labels is noticing that there are some relationships between them, which are captured by the workers' confusion matrices.

Therefore, we can use worker labels along with workers' confusion matrices to infer the most likely true label for each instance, and then figure out how to quantify the uncertainty of each inferred label. Specifically, given a crowdsourced dataset $C = \{(x_i, \mathcal{L}^i)\}_{i=1}^N$, we aim to get a new dataset, denoted by $\mathcal{D} = \{(x_i, y_i', r^{(i)})\}_{i=1}^N$, where y_i' and $r^{(i)}$ represent the inferred label and the uncertainty of the inferred label mentioned above. We will present how to compute y_i' and $r^{(i)}$ in Sections 3.2 and 3.3, respectively.

3.2 Computing Consolidated Labels

Given an instance x_i with worker labels $\mathcal{L}^i$, the basic idea of getting the x_i's most likely label is to compare the values of two conditional probabilities: $P(Y_i = +1 \mid \mathcal{L}^i)$ and $P(Y_i = -1 \mid \mathcal{L}^i)$, the probability that instance x_i has a true label of +1 (resp. -1), conditioned on the labels that the workers provide. If the former (latter) is larger, it means that the instance's label is more likely to be +1 (resp. -1).

We use the work of Dawid and Skene [8] to compute the conditional probabilities. Below is an explanation of their approach, adapted to our notation. Assuming that workers provide labels independently of one another, we have:

$$\begin{aligned} P(Y_i = +1 \mid \mathcal{L}^i) &= \frac{P(\mathcal{L}^i \mid Y_i = +1) P(Y_i = +1)}{P(\mathcal{L}^i)} \\ &\propto P(\mathcal{L}^i \mid Y_i = +1) P(Y_i = +1) \\ &= P(Y_i = +1) \prod_{l_{i,j} \in \mathcal{L}^i} P(L_j = l_{i,j} \mid Y_i = +1) \\ &= P(Y = +1) \prod_{l_{i,j} \in \mathcal{L}^i} q^{(j)}_{+1, l_{i,j}} \end{aligned} \tag{1}$$

Similarly, we can compute $P(Y_i = -1 \mid \mathcal{L}^i)$. By comparing their values, we obtain the consolidated label y_i' (if there is a tie, we break the tie randomly).

$$y_i' = \underset{\bar{y}=-1,1}{\operatorname{argmax}} P(Y = \bar{y}) \prod_{l_{i,j} \in \mathcal{L}^i} q^{(j)}_{\bar{y}, l_{i,j}} \tag{2}$$

Thus, the chance of instance x_i having a ground truth label of, say, +1 is influenced by two main factors. If the instances for which $Y = +1$ are extremely common (i.e. $P(Y = +1)$ is very close to 1), this increases our belief that $Y_i = +1$. Likewise, if the labels that workers provide are likely to happen provided that $Y = +1$ were true (i.e., $q^k_{+1,l}$ is very close to +1), then this also increases our belief that $Y_i = +1$.

3.3 Quantifying Consolidated Label's Uncertainty

Suppose an instance x_i is labeled by a group of k_i workers, denoted by $\mathcal{W}_i$. Let y_i' denote the consolidated label inferred from worker labels using the above method. Like the definition of a worker's confusion matrix, we define the *consolidated confusion matrix* associated with a group of workers as a 2×2 matrix:

$$r^{(\mathcal{W}_i)} = \begin{bmatrix} r^{(\mathcal{W}_i)}_{-1,-1}, & r^{(\mathcal{W}_i)}_{-1,+1} \\ r^{(\mathcal{W}_i)}_{+1,-1}, & r^{(\mathcal{W}_i)}_{+1,+1} \end{bmatrix},$$

where each row represents a true label, each column represents a consolidated label, and $r^{(\mathcal{W}_i)}_{y,y'}$ ($y \in \{\pm 1\}, y' \in \{\pm 1\}$) means that given an instance with true label y, the consolidated label is y' with probability of $r^{(\mathcal{W}_i)}_{y,y'}$. In other words, each cell in the matrix is a

conditional probability $r_{y,y'}^{(\mathcal{W}_i)} = P(Y_i' = y' \mid Y = y)$ $(y', y \in \{\pm 1\})$. If the context is clear, $r^{(\mathcal{W}_i)}$ will be abbreviated as $r^{(i)}$ or r.

We use the Law of Total Probability to compute the conditional probability.

$$P(Y_i' \mid Y) = \sum_n P(Y_i' \mid \bar{\mathcal{L}}_n^i, Y) P(\bar{\mathcal{L}}_n^i \mid Y), \tag{3}$$

where $\{\bar{\mathcal{L}}_n^i : n = 1, 2, \cdots, 2^{k_i}\}$ represents all combinations of the labels that the group of k_i workers from $\mathcal{W}_i$ can provide. For example, suppose there are two workers, w_1, w_2. Then, there will be four combinations of worker labels: $\bar{\mathcal{L}}_1^i = \{(w_1, -1), (w_2, -1)\}$, $\bar{\mathcal{L}}_2^i = \{(w_1, -1), (w_2, +1)\}$, $\bar{\mathcal{L}}_3^i = \{(w_1, +1), (w_2, -1)\}$, and $\bar{\mathcal{L}}_4^i = \{(w_1, +1), (w_2, +1)\}$, where, e.g., $\bar{\mathcal{L}}_1^i$ means that w_1 provides -1 and w_2 provides -1.

The equation depends on two forms of conditional probabilities: $P(Y' \mid \bar{\mathcal{L}}_n^i, Y)$ and $P(\bar{\mathcal{L}}_n^i \mid Y)$. For the latter, we have already discussed how to compute it in Equation 1.

$$P(\bar{\mathcal{L}}_n^i \mid Y) = \prod_{(w_j, l) \in \bar{\mathcal{L}}_n^i} P(\bar{\mathcal{L}}_n^i = (w_j, l) \mid Y) = \prod_{(w_j, l) \in \bar{\mathcal{L}}_n^i} q_{y,l}^{(j)} \tag{4}$$

For the former, since y_i' depends only on $\bar{\mathcal{L}}_n^i$ (i.e., the workers and the labels that they provide), we have that:

$$P(Y' = y' \mid \bar{\mathcal{L}}_n^i, Y) = \begin{cases} 1 & \text{if } y' = \bar{y} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\bar{y}$ represents the consolidated label inferred from $\bar{\mathcal{L}}_n^i$.

4 MODEL EVALUATION

Now we have a noisy dataset $\mathcal{D} = \{(x_i, y_i', r^{(i)})\}_{i=1}^N$, where each instance is associated with a single noisy label y_i' along with a noise rate $r^{(i)}$. There are a number of challenging problems that need to be addressed: (1) how to develop a unified estimation framework that works for different evaluation metrics (Section 4.1); (2) how to bound the difference between the estimated value and the true value under the new framework (Section 4.2); (3) how to give a quantitative analysis on how each factor (e.g., sample vs. population, noisy labels vs. true labels) contributes to the bound (Section 4.2.1). In this section, we propose novel solutions to these non-trivial problems.

4.1 Estimating Model's True Performance

We first present our unified estimation framework. Recall that a model's performance (e.g., accuracy, F-score) is determined by its confusion matrix: $\begin{bmatrix} \mathsf{TN} & \mathsf{FN} \\ \mathsf{FP} & \mathsf{TP} \end{bmatrix}$. To estimate a model's performance, the key is to figure out how to estimate the values of the four cells in the confusion matrix. We will use TP as an example to illustrate this estimation process.

Overview. The estimation framework consists of two steps. The first step is to write TP in a form of the sum of *loss*, and the second step is to use an existing approach [23] to estimate the loss.

Step 1. A loss function, denoted by $\mathsf{Loss}(t, y)$, measures the difference between a model's prediction t and a true label y. When true labels are accessible, we can represent TP as follows:

$$\mathsf{TP} = \sum_{i=1}^N \mathsf{Loss}(t_i, y_i), \tag{6}$$

where $\mathsf{Loss}(t_i, y_i) = 1$ if t_i and y_i are both positive; 0, otherwise.

Step 2. In reality, however, we do not have access to true labels but only noisy labels. Thus, we need to use a noisy label y' along with its noise rate r to estimate the true loss $\mathsf{Loss}(t, y)$. Natarajan et al. [23] proposed an unbiased estimator:

$$\widetilde{\mathsf{Loss}}(t, y') = \frac{(1 - r_{-y',y'}) \cdot \mathsf{Loss}(t, y') - r_{y',-y'} \cdot \mathsf{Loss}(t, -y')}{1 - r_{+1,-1} - r_{-1,+1}}. \tag{7}$$

Please note that this estimator works for any bounded loss function. It is defined based on a noisy label y', thus does not require knowing the true label y.

Unbiased Estimator of TP. By plugging $\widetilde{\mathsf{Loss}}(t, y')$ into Equation 6, we obtain an estimator of TP:

$$\widetilde{\mathsf{TP}} = \sum_{i=1}^N \widetilde{\mathsf{Loss}}(t_i, y_i'). \tag{8}$$

Since $\widetilde{\mathsf{Loss}}(t, y')$ is unbiased, due to the linearity of expectation, we can easily prove that $\widetilde{\mathsf{TP}}$ is unbiased, i.e., $\mathbb{E}[\widetilde{\mathsf{TP}}] = \mathsf{TP}$.

Unbiased Estimator of TN, FN, FP. Using a similar approach, we can get an unbiased estimator for TN, FN, and FP. The only difference from TP is that they need to choose a different loss function. For example, suppose we want to estimate TN. Then, the loss function w.r.t. TN should be defined as $\mathsf{Loss}(t_i, y_i) = 1$ *if* t_i *and* y_i *are both negative; 0, otherwise*.

Estimating Accuracy, Precision, Recall, F-Score. Now we have known how to estimate each value in the model's confusion matrix. These estimated values can be composed to get the model's performance w.r.t. each evaluation metric that is defined in Figure 3. For example, by plugging the estimated values of TP and TN into the accuracy's definition, we can get the estimated value of accuracy: $\frac{\widetilde{\mathsf{TP}} + \widetilde{\mathsf{TN}}}{N}$. Similarly, we can get the estimated values for precision: $\frac{\widetilde{\mathsf{TP}}}{\widetilde{\mathsf{TP}} + \widetilde{\mathsf{FP}}}$, recall: $\frac{\widetilde{\mathsf{TP}}}{\widetilde{\mathsf{TP}} + \widetilde{\mathsf{FN}}}$, and F-score: $\frac{2\widetilde{\mathsf{TP}}}{2\widetilde{\mathsf{TP}} + \widetilde{\mathsf{FP}} + \widetilde{\mathsf{FN}}}$.

For accuracy, we can prove that the estimation is unbiased, i.e.,

$$\mathbb{E}\Big[\frac{\widetilde{\mathsf{TP}} + \widetilde{\mathsf{TN}}}{N}\Big] = \frac{\mathbb{E}[\widetilde{\mathsf{TP}}] + \mathbb{E}[\widetilde{\mathsf{TN}}]}{N} = \frac{\mathsf{TP} + \mathsf{TN}}{N}.$$

However, for the other three evaluation metrics, their estimators are not unbiased. This is because that when both numerator X and denominator Y are random variables, we do *not* have $\mathbb{E}[\frac{X}{Y}] = \frac{\mathbb{E}[X]}{\mathbb{E}[Y]}$. This type of estimator is often called conditionally unbiased given Y. Despite that they are not unbiased in theory, we validate their effectiveness in the experiments, and find that they perform very well on both synthetic and real-world datasets.

4.2 Bounding Estimation Error

In this section, we study how to compute confidence intervals for these estimators. This problem is challenging because there are two sources of error involved and a confidence interval has to take both of them into consideration.

Sample vs. Population. The first source of error comes from sampling. Since the entire population is not accessible, our estimator can only look at a sample of data and use it to estimate how well a model will perform over the entire population.

Noisy Labels vs. True Labels. The other source of error comes from noisy labels. Since true labels are not accessible, our estimator can only look at noisy labels and use them to estimate how well a model will perform w.r.t. true labels.

To address this challenge, we develop an analytical confidence interval based on the central limit theorem (CLT). From the analytical confidence interval, we can easily see how each source of error contributes to the overall estimation error (i.e., half the width of the confidence interval). However, the analytical confidence interval only works for accuracy. For the other evaluation metrics, we show how to compute their empirical confidence intervals using bootstrapping and conduct experiments to explore the impact of the two sources of error on the overall estimation error in various situations.

4.2.1 Analytical Confidence Interval. We first introduce some background knowledge about CLT, then present an analytical confidence interval for accuracy, and finally dive into the confidence interval to gain more insights.

Central Limit Theorem (CLT). Consider a population with mean μ and variance σ^2. Given a random sample of size N from the population, $\{X_1, X_2, \cdots, X_N\}$, CLT states that the sample mean $\tilde{\mu} = \frac{1}{N}\sum_{i=1}^{N} X_i$ follows a normal distribution with mean μ and variance $\frac{\sigma^2}{N}$. Note that CLT does not require that the original population has to follow a normal distribution.

Suppose that we treat the sample mean as an estimator of the population mean. Based on CLT, the confidence interval for the estimator is

$$\mu \pm \lambda\sqrt{\frac{\sigma^2}{N}}, \tag{9}$$

where λ is a parameter determined by a confidence level (e.g., $\lambda = 1.96$ for 95% confidence interval, $\lambda = 2.58$ for 99% confidence interval). Since the population mean μ and variance σ^2 are unknown, they can be replaced by the estimated mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$ based on a sample.

Analytical Confidence Interval. The estimator of accuracy can be represented as follows:

$$\text{accuracy} \approx \frac{\widetilde{\text{TP}} + \widetilde{\text{TN}}}{N} = \frac{1}{N}\sum_{i=1}^{N} \text{Loss}_{0/1}(t_i, y_i'),$$

where $\widetilde{\text{Loss}}_{0/1}(t_i, y_i')$ is an unbiased estimator of $\text{Loss}_{0/1}(t_i, y_i)$, and $\text{Loss}_{0/1}(t_i, y_i) = 1$ if t_i and y_i have the same sign (i.e., either both positive or both negative); $\text{Loss}_{0/1}(t_i, y_i) = 0$, otherwise. We can see that the estimator is in the form of mean. Let $X_i = \widetilde{\text{Loss}}_{0/1}(t_i, y_i')$ for each $i \in [1, N]$. The confidence interval for the estimator can be directly derived from Equation 10.

$$\mathbb{E}[X] \pm \lambda\sqrt{\frac{\text{var}(X)}{N}} \tag{10}$$

In-Depth Analysis. We now provide an in-depth analysis of the confidence interval. As mentioned in the beginning of this section, there are two sources of error. Our analysis aims to answer two questions: (1) *how does sample size affect the confidence interval?* (2) *how does label noise affect the confidence interval?*.

For simplicity, we assume that each instance has the same label noise of $r_{-1,+1} = r_{+1,-1} = \beta$. Based on Equation 7, we find that $\widetilde{\text{Loss}}_{0/1}(t, y')$ can only take two possible values:

$$\widetilde{\text{Loss}}_{0/1}(t, y') = \begin{cases} \frac{1-\beta}{1-2\beta} & \text{if } t \text{ and } y' \text{ have } \textit{the same} \text{ sign} \\ \frac{-\beta}{1-2\beta} & \text{otherwise} \end{cases} \tag{11}$$

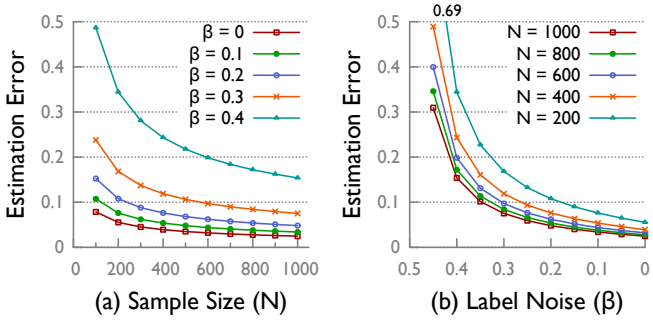


Figure 4: The relationships between sample size (N), label noise (β), and estimation error (half the width of the 95% confidence interval), for model accuracy $\theta = 0.8$. (For simplicity, we do not show the estimation error for $\beta \in (0.5, 1]$ because based on Equation 15, it will be the same as $1 - \beta$).

Suppose that $\widetilde{\text{Loss}}_{0/1}(t, y')$ has a probability of p being $a = \frac{1-\beta}{1-2\beta}$ and $1 - p$ being $b = \frac{-\beta}{1-2\beta}$. It is easy to see that $a + b = 1$. The expected value of $\widetilde{\text{Loss}}_{0/1}(t, y')$ is:

$$\mathbb{E}[X] = pa + (1-p)b \tag{12}$$

The variance of $\widetilde{\text{Loss}}_{0/1}(t, y')$ is:

$$\begin{aligned} \text{Var}[X] &= \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ &= pa^2 + (1-p)b^2 - \mathbb{E}[X]^2 \\ &= pa^2 + (1-p)b^2 - \Big(pa + (1-p)b - \mathbb{E}[X]\Big) - \mathbb{E}[X]^2 \\ &= -ab + \mathbb{E}[X] - \mathbb{E}[X]^2 \end{aligned} \tag{13}$$

Let θ denote a model's true accuracy. Since $\frac{1}{N}\sum\widetilde{\text{Loss}}_{0/1}(t, y')$ is an unbiased estimator of the true accuracy, then we have $\mathbb{E}[X] = \theta$.

$$\text{Var}[X] = -ab + \theta - \theta^2 \tag{14}$$

By plugging Equation 14 into Equation 10, we obtain a closed form confidence interval of our estimator:

$$\theta \pm \lambda\sqrt{\frac{-ab + \theta - \theta^2}{N}} \tag{15}$$

where $ab = \frac{-\beta(1-\beta)}{(1-2\beta)^2}$, β is label noise, θ is a model's true accuracy, N is sample size, and λ is constant determined by a confidence level.

From this equation, we can analyze how each source of error contributes to the overall estimation error.

Insight 1. The first source of error (sample vs. population) is controlled by sample size N. It only affects the denominator of the confidence interval. Figure 4(a) shows the relationship between sample size and estimation error, for different label noise β. We can see that as sample size N increases, regardless of what β is, estimation error will decrease at a rate of $O\left(\frac{1}{\sqrt{N}}\right)$. For example, when sample size is increased from $N = 100$ to 1000, estimation error will decrease by about $O\left(\sqrt{\frac{1000}{100}}\right) = 3$ times.

Insight 2. The second source of error (noisy label vs. true label) is controlled by label noise β. It only affects the numerator of the confidence interval. Figure 4(b) demonstrates the relationship between label noise and estimation error, for different sample size N. We can see that as noise decreases, regardless of what N is, estimation

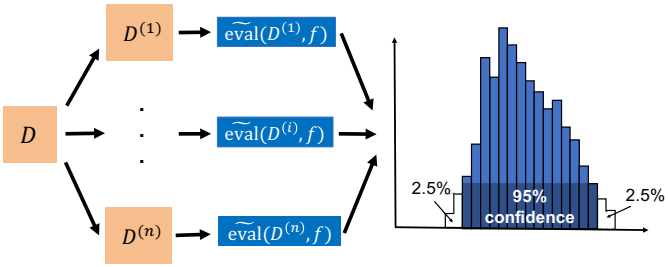


Figure 5: An illustration of using bootstrap to compute empirical confidence intervals for complex estimators.

error will decrease at a rate of $O\left(\frac{1}{|0.5-\beta|}\right)$. For example, when β is decreased from $\beta = 0.4$ to $\beta = 0.1$, estimation error will decrease by about $O\left(\frac{|0.5-0.1|}{|0.5-0.4|}\right) = 4$ times.

4.2.2 Empirical Confidence Interval. We first introduce some background knowledge about bootstrapping, and then present our approach to compute empirical confidence intervals for other estimators than accuracy.

Bootstrapping. Consider a population P and a random sample of the population S. Given an estimator est, suppose the estimator can use the sample to estimate a *complex* parameter of the population. Once the estimate $\text{est}(S)$ is derived, we want to compute its confidence interval. If we could create multiple samples from the population, $S_1, S_2, \cdots, S_n$, then we would get a distribution of the estimates, $\text{est}(S_1), \text{est}(S_2), \cdots, \text{est}(S_n)$. Based on the distribution, we can find an interval that covers 95% of the estimates and use it as a 95% confidence interval of the estimation. In practice, however, the population is not available, thus it is impossible to draw another sample from P. The basic idea of bootstrapping is to simulate this procedure by constructing a number of resamples of S. Specifically, to get a resample, it samples the data from S with replacement. Let $S^{(1)}, S^{(2)}, \cdots, S^{(n)}$ denote n resamples. Then, it can get a distribution of the estimates based on resamples, $\text{est}(S^{(1)}), \text{est}(S^{(2)}), \cdots, \text{est}(S^{(n)})$, from which we can compute a 95% confidence interval.

Empirical Confidence Interval. Now we present how to use the bootstrap to compute empirical confidence intervals for our estimators. Consider a crowdsourced dataset D. We can think of D as a random sample of a population. While it is feasible to get multiple crowdsourced datasets from the population, the monetary cost and the time for doing so can be quite high. For example, suppose each instance needs 0.5 dollar and 5 seconds to label on average. Getting one crowdsourced dataset of size $|D| = 1000$ will cost us \$500 dollars and 1.4 hours. Repeat this for 1000 times will cost us as high as 0.5 million dollars and 58 days.

We use boostrapping to avoid the need to repeatedly draw samples from the population. Figure 5 illustrates the procedure. Given D and an estimator $\widetilde{\text{eval}}(D, f)$, the first step is to construct n resamples of D, denoted by $D^{(1)}, D^{(2)}, \cdots, D^{(n)}$. Then, we apply the estimator to each resample to get n estimates,

$$\widetilde{\text{eval}}(D^{(1)}, f), \quad \widetilde{\text{eval}}(D^{(2)}, f), \quad \cdots, \quad \widetilde{\text{eval}}(D^{(n)}, f).$$

Given a confidence level (e.g., 95%), let $\widetilde{\text{eval}}_{2.5\%}$ and $\widetilde{\text{eval}}_{97.5\%}$ denote the 2.5th and 97.5th percentile of the distribution, respectively. Then, the 95% confidence interval is denoted by $[\widetilde{\text{eval}}_{2.5\%}, \widetilde{\text{eval}}_{97.5\%}]$.

This approach works for all the estimators developed in Section 4.1 including precision, recall, F-score. Like accuracy, the estimation error of other estimators also come from two sources. We empirically study its relationship with sample size and label noise for these estimators in the experiments.

5 CLEANING STRATEGY

Now we present how TARS provides the second piece of advice: which label should be cleaned? Please note that, unlike the previous section, here we turn our focus to the *model training* stage.

Typically, in supervised learning, we are given $S = \{(x_i, y_i)\}_{i=1}^N$ drawn from $\mathbb{G}$, and aim to make predictions on the dataset $S_{\text{test}} = \{(x_j, y_j)\}_{j=1}^M$, also drawn from $\mathbb{G}$. We could train a model f on S, predict labels on instances x_j from S_{test}, and compare the predicted labels with each y_j. Ideally, the predicted labels should be "fairly close" to the actual labels S_{test}. One reason why this procedure works is because both S and S_{test} are drawn from $\mathbb{G}$; in other words, the data we train on is an acceptable representation of the data we are expected to make predictions on.

Labels in noisy dataset $\mathcal{D} = \{(x_i, y_i', r^{(i)})\}_{i=1}^N$ are not guaranteed to be correct, so $\mathcal{D}$ might not adequately represent the data we ultimately have to make predictions on. This means if we train directly on the pairs $\{(x_i, y_i')\}_{i=1}^N$ and are asked to predict the labels from a testing set S_{test}, there could be unacceptably many errors. Therefore, we study the cleaning strategy problem, aiming to choose the "best" instance to clean that would bring the greatest benefit to the resulting model.

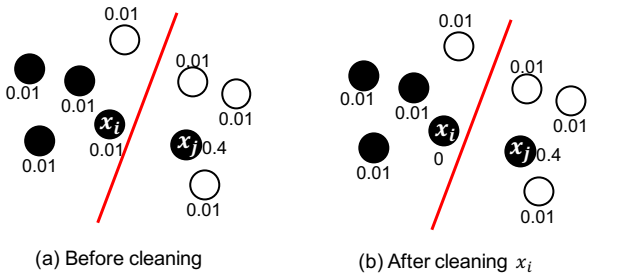
In the following, we first explain why the existing cleaning strategies do not work in Section 5.1, and then present the main idea of our cleaning strategy in Section 5.2. We find that the naive implementation of this idea did not work very well in the experiments. We discuss the issues and propose effective solutions in Section 5.3.

5.1 Limitations of Existing Cleaning Strategies

Below are three classes of existing cleaning strategies, and explanations as to why they are not the perfect solution to our problem.

Active Learning. Typical active learning settings start with a small pool of (cleanly) labeled data, and a large pool of unlabeled data. Similar to our problem setting, an oracle is available to obtain ground truth labels, and the goal is to choose the most informative instances to label under a budget [29]. Active learning literature refers to approaches to this problem as query strategies. There are many query strategies proposed in the literature, such as uncertain sampling [17], expected error reduction [28]. However, since active learning involves labeling *unlabeled* rather than *noisy* data, we believe that an effective cleaning strategy in our problem setting should leverage y_i' and $r^{(i)}$, rather than treat instances as unlabeled.

Consider an example in Figure 6, where black (white) points represent positive (negative) instances and the red line represents the model. Since labels are noisy, there is a number associated with each point representing its label's noise rate. If we ignore noise rates and simply apply an active-learning query strategy (e.g., uncertain sampling), x_i will be selected because it is closest to the model's decision boundary. However, x_i's label only has a noise rate of 0.01, which is very unlikely to flip after cleaning. As shown in Figure 6 (b), if x_i's label did not flip, the model would keep unchanged, thus it is a waste of cleaning budget. In comparison, x_j has a much higher noisy rate, and cleaning it would be more likely to flip the label, leading to a big change of the model. Thus, x_j should have a higher priority than x_i to be selected, for this example.



(a) Before cleaning (b) After cleaning x_i

Figure 6: An illustration of the limitation of uncertain sampling.

Sorting by Noise Rate. To overcome the limitation, one possible strategy could involve cleaning instances whose labels are most likely to be wrong: sort the instances by noise rates, then choose the instance with the greatest noise. This minimizes the chance of *not* flipping an instance's label after cleaning, but it totally ignores the impact to the model. If the model does not have much change, it will not be necessary to clean an instance's label even if the instance's label is flipped.

Consider an example in Figure 7. The above cleaning strategy will select x_i for cleaning because it has the greatest noise. As shown in Figure 7(b), even if x_1's label was flipped, the model would still keep unchanged. In comparison, x_j's label has a slightly smaller noise rate, but if its label was flipped, the model's decision boundary would move from the left side of x_j to its right side, leading to a big change of the model. Therefore, x_j should have a higher priority to be selected, for this example.

ActiveClean. As we can see, a good cleaning strategy should be aware of both noise rate as well as model changes. A recent paper takes these two factors into consideration and proposes a new cleaning strategy, called ActiveClean [16]. ActiveClean predicts the true label of each instance, and then estimates how cleaning each instance would change the model based on predicated true labels, and finally selects the instance that would lead to the biggest change. However, ActiveClean has two limitations to solve our problem. First, it does not leverage the given noise rates to predict the true label of each instance. Second, it uses stochastic gradient descent to update the model after cleaning each batch of instances, thus the model's performance may not be very stable for our oracle-based cleaning scenario, where only a small number of instances (e.g., hundreds of instances) can be cleaned.

5.2 Main Idea: Expected Model Improvement

To inform our decision of which instance to clean, we wish to understand how cleaning an instance could improve the current model. Let f be the model trained without cleaning any data. Let f_i be the model trained after cleaning instance x_i. There are two possible cases about f_i.

Case 1: If the label is *not* flipped (i.e., $y_i = y_i'$), the model will stay the same, i.e., $f_i = f$.

Case 2: If the label is flipped (i.e., $y_i = -y_i'$), the model's performance will change by $\mathsf{eval}(\mathbb{G}, f_i) - \mathsf{eval}(\mathbb{G}, f)$.

Please note that cleaning an instance is not always guaranteed to improve the model. For example, in Case 2, if $\mathsf{eval}(\mathbb{G}, f_i) < \mathsf{eval}(\mathbb{G}, f_i)$, the model's performance will get worse. Since we do not know whether the label would be flipped or not until cleaning the instance, the model's true improvement cannot be obtained. Nevertheless, it could be possible to compute the model's expected improvement based on the probabilities that each case may happen.

(a) Before cleaning (b) After cleaning x_i

Figure 7: An illustration of the limitation of sorting by noise rate.

Let $P(\mathsf{case}_1)$ and $P(\mathsf{case}_2)$ denote the probabilities that Case 1 and Case 2 happen, respectively. Then, the *expected model improvement* (EMI) is defined as:

$$\begin{aligned}\mathsf{EMI}(i) &= P(\mathsf{case}_1)\cdot 0 + P(\mathsf{case}_2)\cdot\Big(\mathsf{eval}(\mathbb{G}, f_i) - \mathsf{eval}(\mathbb{G}, f)\Big)\\ &= P(\mathsf{case}_2)\cdot\Big(\mathsf{eval}(\mathbb{G}, f_i) - \mathsf{eval}(\mathbb{G}, f)\Big)\end{aligned} \tag{16}$$

Our cleaning strategy computes $\mathsf{EMI}(i)$ for each instance x_i, and then selects the instance x_{i^*} with the largest value and sends it to an oracle to clean.

$$i^* = \operatorname*{argmax}_i \ \mathsf{EMI}(i) \tag{17}$$

Next, we discuss how to compute $\mathsf{EMI}(i)$, which consists of three parts:

Computing $P(\mathsf{case}_2)$. $P(\mathsf{case}_2)$ represents the probability that the noisy label is flipped after cleaning. Note that we have already known that the noisy label is y_i'. Thus, $P(\mathsf{case}_2)$ represents the conditional probability of the true label being $-y_i'$ given the noisy label y_i', i.e., $P(\mathsf{case}_2) = P(Y_i = -y_i'|Y_i' = y_i')$, where $y_i' \in \{-1, +1\}$ is constant. Based on the Bayes' rule, we can easily derive that

$$\begin{aligned}P(\mathsf{case}_2) &= \frac{P(Y_i' = y_i'|Y_i = -y_i')\cdot P(Y_i = -y_i')}{P(Y_i' = y_i')}\\ &\propto P(Y_i' = y_i'|Y_i = -y_i')\cdot P(Y_i = -y_i')\end{aligned} \tag{18}$$

$P(Y_i' = y_i'|Y_i = -y_i')$ is equal to the noise rate $r^{(i)}_{-y_i', y_i'}$ and $P(Y_i = -y_i')$ is equal to the prior $P(Y = -y_i')$. Therefore, we obtain

$$P(\mathsf{case}_2) \propto r^{(i)}_{-y_i', y_i'}\cdot P(Y = -y_i') \tag{19}$$

Similarly, we can obtain

$$P(\mathsf{case}_1) \propto r^{(i)}_{y_i', y_i'}\cdot P(Y = y_i') \tag{20}$$

Since $P(\mathsf{case}_1) + P(\mathsf{case}_2) = 1$, we have

$$P(\mathsf{case}_2) = \frac{r^{(i)}_{-y_i', y_i'}\cdot P(Y = -y_i')}{r^{(i)}_{-y_i', y_i'}\cdot P(Y = -y_i') + r^{(i)}_{y_i', y_i'}\cdot P(Y = y_i')} \tag{21}$$

Computing $\mathsf{eval}(\mathbb{G}, f)$. Since $\mathbb{G}$ is not available, we cannot compute $\mathsf{eval}(\mathbb{G}, f)$ directly. Fortunately, in Section 4.1, we have discussed a way to estimate it based on noisy data $\mathcal{D}$, which is accessible. Thus, we can use $\widetilde{\mathsf{eval}}(\mathcal{D}, f)$ to approximate $\mathsf{eval}(\mathbb{G}, f)$.

Computing $\mathsf{eval}(\mathbb{G}, f_i)$. If we knew f_i, $\mathsf{eval}(\mathbb{G}, f_i)$ could be estimated similarly as above. Recall that f_i denotes the resulting model from training on D after cleaning instance x_i. If x_i's label is not flipped, we do not need to consider this case because the model stays the same as f; if x_i's label is flipped, we can retrain a model f_i on the new dataset $\mathcal{D}_i$, where $\mathcal{D}_i$ represents the dataset resulting from flipping the label of instance x_i of $\mathcal{D}$.

Remarks. EMI incorporates both noise rates and model changes, thus overcomes the limitations of the existing cleaning strategies. For example, consider x_i in Figure 6. Since it has a small noise

rate, leading to a small value of $P(\text{case}_2)$, EMI tends to not select x_i. Consider x_i in Figure 7. Since the model would not change after flipping the label of x_i, leading to a zero value of $\text{eval}(\mathbb{G}, f_i) - \text{eval}(\mathbb{G}, f)$, EMI will not select x_i.

However, the cost of having this effective cleaning strategy is that it needs to retrain N models, $f_1, f_2, \cdots, f_N$, for each iteration, which could be highly inefficient when $\mathcal{D}$ is large. Fortunately, EMI is similar in spirit to an active learning query strategy called *expected error reduction* [28], which also needs to retrains N models. Many techniques have been proposed to reduce the retraining time, such as incremental training and subsampling techniques. In this paper, we treat this as an orthogonal problem and defer additional exploration to future work.

5.3 Further Optimization

We find that the naive implementation of EMI did not perform very well in the experiments. We discuss the reasons that cause the problem and propose effective techniques to optimize EMI.

Splitting the noisy data The first reason is related to which noisy dataset should be used to estimate $\text{eval}(\mathbb{G}, f_i)$ and $\text{eval}(\mathbb{G}, f)$. One natural idea is to use $\mathcal{D}_i$ for $\text{eval}(\mathbb{G}, f_i)$ and $\mathcal{D}$ for $\text{eval}(\mathbb{G}, f)$ because f_i is trained on $\mathcal{D}_i$ and f is trained on $\mathcal{D}$:

$$\text{eval}(\mathbb{G}, f_i) \approx \widetilde{\text{eval}}(\mathcal{D}_i, f_i), \quad \text{eval}(\mathbb{G}, f) \approx \widetilde{\text{eval}}(\mathcal{D}, f).$$

However, there are two issues about this idea.

First, as shown in Equation 16, the goal is to estimate the difference between $\text{eval}(\mathbb{G}, f_i)$ and $\text{eval}(\mathbb{G}, f)$ as more accurate as possible. Let X and Y denote the estimators of $\text{eval}(\mathbb{G}, f_i)$ and $\text{eval}(\mathbb{G}, f)$, respectively. That is, we aim to minimize $\text{var}(X - Y) = \text{var}(X) + \text{var}(Y) - \text{cov}(X, Y)$. In order to minimize $\text{var}(X - Y)$, we need to increase $\text{cov}(X, Y)$ as more as possible, i.e., making X and Y as more correlated as possible. If X and Y are estimated based on the same noisy data, it will make them much more correlated than be estimated on two different ones. Another issue is about overfitting. If we train a model on a dataset and then use the same dataset to evaluate it, the model may suffer from overfitting. In other words, the model may perform well on the current dataset, but not learn to generalize to unseen data.

To address these issues, we split $\mathcal{D}$ into a training dataset $\mathcal{D}_{\text{train}}$ and a validation dataset $\mathcal{D}_{\text{vdn}}$. Only the instances in $\mathcal{D}_{\text{train}}$ can be cleaned and be used to train a model; the instances in $\mathcal{D}_{\text{vdn}}$ cannot be cleaned or train a model, and their job is to estimate $\text{eval}(\mathbb{G}, f_i)$ and $\text{eval}(\mathbb{G}, f)$:

$$\text{eval}(\mathbb{G}, f_i) \approx \widetilde{\text{eval}}(\mathcal{D}_{\text{vdn}}, f_i), \quad \text{eval}(\mathbb{G}, f) \approx \widetilde{\text{eval}}(\mathcal{D}_{\text{vdn}}, f).$$

It is worth mentioning that this idea has been widely adopted in machine learning, where a validation set is often used for hyperparameter tuning and has shown to be very effective to avoid overfitting.

Weighing with Model Uncertainty Even after splitting noisy data into validation and training sets, another challenge that remains is that the values for $\text{EMI}(i)$ for different instances i can be very similar. To see why this is the case, consider a simple situation where $\text{eval}(\mathbb{G}, f)$ measures the percent of instances classified correctly by f, and the data is labeled by a single worker. Then, the $r^{(i)}$ constant across all instances i, as is $P(\text{case}_2)$.

This means if we have two instances i_1 and i_2, for which $\text{eval}(\mathbb{G}, f_{i_1}) - \text{eval}(\mathbb{G}, f)$ and $\text{eval}(\mathbb{G}, f_{i_2}) - \text{eval}(\mathbb{G}, f)$ are very similar, the resulting values $\text{EMI}(i_1)$ and $\text{EMI}(i_2)$ will be very similar. In the worst case, $\text{EMI}(i_1) = \text{EMI}(i_2)$, which makes impossible to distinguish which instance would be a better candidate for cleaning.

Table 1: Dataset statistics (SS: synthetic data with simulated noisy labels; RS: real-world data with simulated noisy labels; RR: real-world data with real-world crowdsourced noisy labels

Dataset	#Positive	#Negative	#Dimension	Type
Gaussian	500	500	2	SS
Heart	120	150	13	RS
German	300	700	20	RS
Cancer	77	168	9	RS
Restaurant	102	1902	8	RR

To address this issue, we combine EMI with the uncertainty of model f. More specifically, let $u(x_i) = 1 - P(f(x_i) \mid x_i)$ measure the uncertainty of f's prediction on x_i. For example, we can interpret small $P(f(x_i) \mid x_i)$ as a "less confident" prediction, which corresponds to large $u(x_i)$.

Intuitively, if we have two instances whose EMI values are similar, we would like to defer the decision of "which is better" (i.e. which instance is better to clean) to the model's uncertainty. In this case, the instance for which f more uncertain (i.e. larger $u(x_i)$) should be considered a better candidate for cleaning.

At first glance, it might be tempting to compare instances using $u(x_i) \cdot \text{EMI}(i)$. Suppose x_{i^*} is the best instance to clean. Since $\text{EMI}(i^*)$ is computed using estimators, it's possible that $\text{EMI}(i^*) < 0$. Furthermore, if $u(x_{i^*})$ is large, then the product $u(x_{i^*}) \cdot \text{EMI}(i^*)$ could be very negative, which would rank x_{i^*} below other instances.

Before multiplying by $u(x_i)$, we need to transform $\text{EMI}(i)$ into a positive value, using some function $\sigma : \mathbb{R} \to \mathbb{R}^+$. In order to preserve the relative ordering of $\text{EMI}(i)$, σ needs to be monotonically increasing. A convenient choice is the sigmoid function $\sigma(t) = \frac{1}{1+e^{-t}}$. To weigh EMI with model uncertainty, we compute:

$$\text{MU}(i) = u(x_i) \cdot \sigma(\text{EMI}(i)), \tag{22}$$

Thus, with this optimization, we choose instance i^* to clean by computing:

$$i^* = \underset{i}{\text{argmax}} \ \text{MU}(i) \tag{23}$$

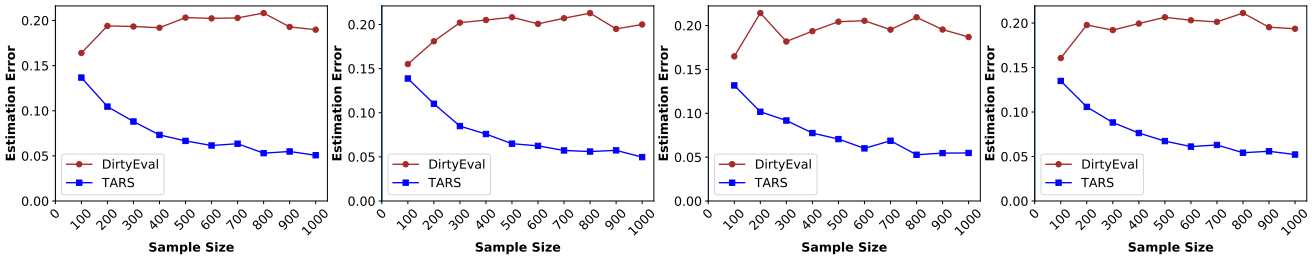
6 EXPERIMENTS

We conduct extensive experiments to evaluate the effectiveness of TARS on synthetic and real-world datasets with simulated noisy labels and crowdsourced noisy labels. The experiments aim to answer four questions. (1) Can we accurately estimate a model's true performance from noisy labels? (2) How does the estimation error change by varying different parameters (e.g., sample size, noise rates)? (3) Are the proposed optimization techniques effective for EMI? (4) How does the optimized EMI perform compared to the existing cleaning strategies?

6.1 Experimental Settings

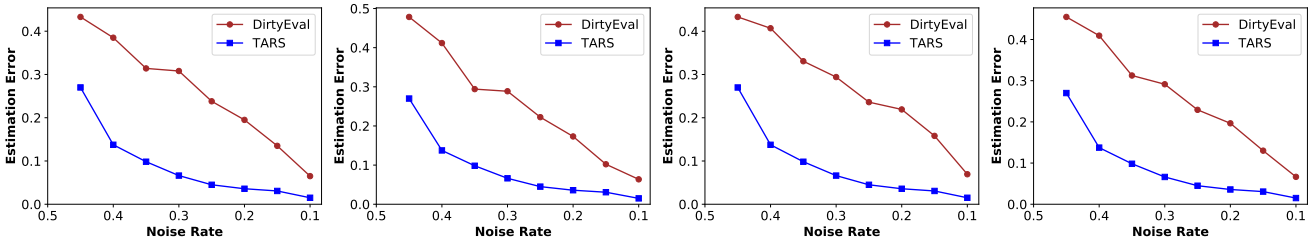
Datasets. We used a synthetic dataset and four real-world datasets to evaluate our method. (1) Gaussian contains instances randomly drawn from two different 2D Gaussian distributions with the parameters of $(x_1, y_1) = (0, 0)$ and $\sigma_{x_1} = \sigma_{y_1} = 1$, and $(x_2, y_2) = (1, 0)$ and $\sigma_{x_2} = \sigma_{y_2} = 1$. (2) Heart, German, and Cancer are three real-world datasets downloaded from the UCI Machine Learning Repository[3]. They are widely used to evaluate classification algorithms in the Machine Learning community. (3) Restaurant is a real-world dataset widely used to evaluate entity resolution. We got the dataset from the authors in [33]. Table 1 illustrates the detailed statistical information of the five data sets.

[3] http://archive.ics.uci.edu/ml/index.php



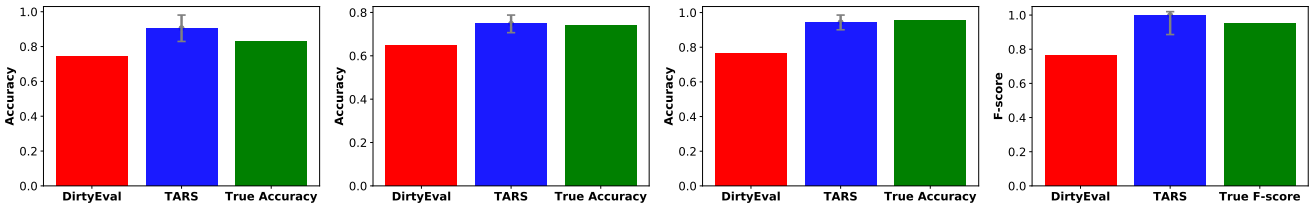
(a) accuracy (b) precision (c) recall (d) F-score

Figure 8: Comparison of the estimation error between TARS and DirtyEval by varying sample size (Gaussian).



(a) accuracy (b) precision (c) recall (d) F-score

Figure 9: Comparison of the estimation error between TARS and DirtyEval by varying noise rates (Gaussian).



(a) **Heart** (b) **German** (c) **Cancer** (d) **Restaurant**

Figure 10: Comparison of the estimation error of TARS and DirtyEval on real-world datasets.

Noisy Labels. The noisy labels for the Gaussian, Heart, German, and Cancer datasets were randomly generated, controlled by two parameters, $r_{-1,+1}$ and $r_{+1,-1}$. For example, given $r_{-1,+1} = 0.2$ and $r_{+1,-1} = 0.1$, to generate the noisy labels for a dataset, we do the following for each instance. If the instance's true label is -1, then it will be flipped with a probability of 0.2; if its true label is +1, then it will be flipped with a probability of 0.1. When an instance needs to be labeled by multiple workers, we will apply this process to generate multiple noisy labels for the instance and get its consolidated label using the method described in Section 3.

The noisy labels for the Restaurant dataset were collected from the real-world crowd workers in Amazon Mechanical Turk (AMT). Each instance was labeled by a single worker, and the entire dataset was labeled by 24 different workers in total. The noise rates of the workers were in the ranges of $r_{-1,+1} \in [0, 0.4]$ and $r_{+1,-1} \in [0, 0.2]$.

Cleaning Strategies. We compared TARS with three existing cleaning strategies. SortNoise cleans the instance whose noisy label is most likely to be wrong, without considering the impact of cleaning the instance to the current model. ActiveClean cleans the instance which, if the instance was cleaned, would impart the greatest change to the current model. Expected Error Reduction (ExpectError) [28] cleans the instance such that the current model's error can be reduced the most, where the error is computed on noisy labels rather than estimated w.r.t. true labels. We chose ExpectError because it is similar (in spirit) to our strategy and has been shown to outperform other active-learning query strategies such as uncertain sampling [28].

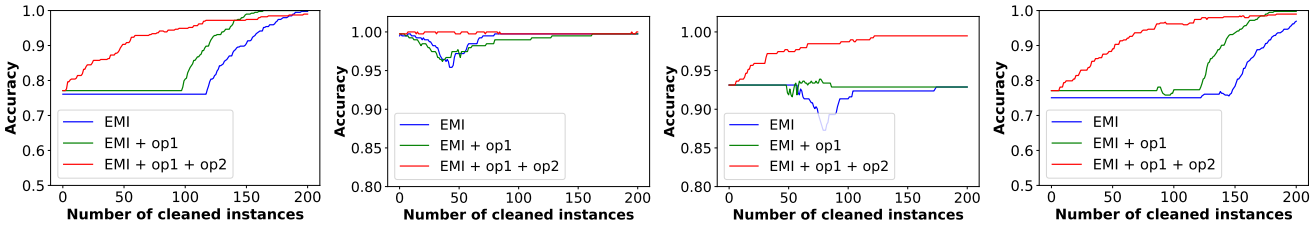
All the code was written in Python 2.7. We trained logistic regression models on all datasets using scikit-learn[4]. Each dataset was randomly divided into a training set and a test set with the ratio of 2 to 1. We ran each experiment ten times and reported the average performance.

6.2 Evaluation of Advice 1

In this section, we first conduct sensitivity analysis on the Advice 1 provided by TARS in order to gain a deep understanding of its performance, and then examine its performance on real-world datasets.

6.2.1 Sensitivity Analysis. We evaluate the estimation error of TARS on the Gaussian dataset by varying the sample size, the noise rate, the number of votes, and the percentage of good workers, for accuracy, precision, recall, and F-score. When varying one parameter, we set the other parameters with their default values. By default, the sample size is 1000, the noise rate is 0.2, the number of votes is 1, and the percentage of good workers is 0%. We define the estimation error of TARS as half the width of the 95% confidence interval of its estimated value. We compared TARS with DirtyEval, which is a naive estimator presented in the Introduction section. It simply treats the noisy labels as the true labels without considering noise rates, thus leading to biased estimated results.

[4]http://scikit-learn.org/



(a) Noise Rates: (0.4, 0) (b) Noise Rates: (0.2, 0.2) (c) 50%:(0.4, 0), 50%:(0.2, 0.2) (d) 90%: (0.4, 0), 10%: (0.2, 0.2)

Figure 11: Evaluating the effectiveness of the proposed optimization techniques of the TARS's cleaning strategy (Gaussian).

Sample Size. Figure 8 compares the estimation error of TARS and DirtyEval by varying the dataset size from 100 to 1000, w.r.t. different evaluation metrics. We can see that as the sample size was increased by 10 times, the estimation error of TARS was reduced by about 3 times, but DirtyEval did not change so much. The reason is that TARS takes into account noise rates, leading to larger variance. Increasing the sample size can reduce the estimator's variance at the rate of $O(\frac{1}{\sqrt{N}})$. In comparison, DirtyEval has a much larger bias but with a smaller variance. While the variance can still be decreased, since the overall estimation error is dominated by the bias, the improvement is marginal.

Noise Rate. Figure 9 compares the estimation error of TARS and DirtyEval by varying the noise rate from 0.45 to 0.1, w.r.t. different evaluation metrics. We can see that TARS follows a similar trend for all the figures: as the decrease of the noise rate, the estimation error will first decrease dramatically, quickly reaching an estimation error of less than 0.1 at the noise rate of about 0.35. After that, the decreasing speed tends to get slower. Our theoretical analysis in Section 4.2.1 has shown that for accuracy, the estimation error decreases at a rate of $O(\frac{1}{|0.5-\beta|})$. This experiment validated that it holds for the other evaluation metrics empirically.

Number of Votes and Percentage of Good Workers. We evaluated the performance of TARS by varying the number of votes and the percentage of good workers, respectively. The results can be found in Appendix A

6.2.2 Performance on Real-world Datasets. We evaluate TARS on the four real-world datasets. We used the noise rate of 0.8 for Heart, German, and Cancer, and used the real-world crowd workers to label Restaurant. We aim to answer two questions in this experiment: (i) can TARS get an accurate estimate of the model's true performance on real-world datasets? and (ii) can TARS bound the difference between the estimated value and the true value?

We used TARS and DirtyEval to estimate the model's true accuracy on the Heart, German, and Cancer datasets. We estimated F-score for the Restaurant dataset because the class labels of the dataset is highly imbalanced. As a comparison, we computed the true accuracy (f-score) using the ground-truth labels. Note that since the entire clean population $\mathbb{G}$ is not available, we can only compute the true accuracy (F-score) based on the clean sample S (i.e., the labeled datasets). Figure 10 shows the result. We can see that on the German and Cancer datasets, TARS returned almost the same accuracy as the true accuracy. On the Heart dataset, although the TARS's performance was not as good as the one on the German and Cancer datasets, it was still better than DirtyEval. On the Restaurant dataset, TARS returned an estimated value of F-score with the error more than 2× smaller than DirtyEval. More importantly, on all the datasets, TARS was able to bound the estimation error. Being able to do so is essential for enabling reliable decisions in the real world.

6.3 Evaluation of Advice 2

In this section, we first examine how the proposed optimization techniques can improve the effectiveness of the naive implementation of EMI, and then compared TARS (i.e., EMI with all the optimization techniques) with the state-of-the-art cleaning strategies, ActiveClean, ExpectError, and SortNoise.

6.3.1 Optimization Techniques. In Section 5.3, we identified the possible issues when applying EMI in practice, and proposed two optimization techniques, denote by op1 and op2, to address them, where op1 represents the optimization of splitting the noisy data and op1 represents the optimization of weighing with model uncertainty. Figure 11 compares the three variants of EMI on the Gaussian dataset in four settings of noise rates.

In Figure 11(a), we set the noise rates to $r_{-1,+1} = 0.4$ and $r_{+1,-1} = 0.2$. We can see that the noisy labels had a significant negative impact on the model. After cleaning 100 instances (10% of the data), EMI +op1+op2 improved the model's accuracy from 0.77 to 0.94. However, the other two cleaning strategies did not help to improve the model so much.

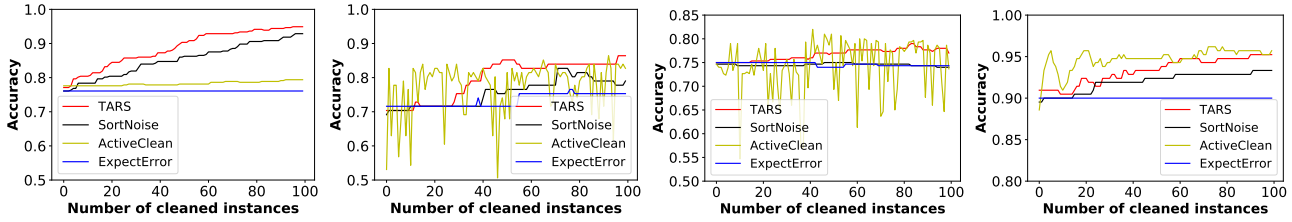
In Figure 11(b), we set the noise rates to $r_{-1,+1} = 0.2$ and $r_{+1,-1} = 0.2$. In this setting, we can see that the model's accuracy was (almost) not affected by the noisy labels. Therefore, all three cleaning strategies started with a very accurate model. After sending some instances to an oracle to clean, EMI and EMI +op1 sometimes led to a much worse model, but EMI +op1+op2 avoided this kind of situation happen.

In Figure 11(c) and (d), we evaluated the optimization techniques on a mix of noise rates, where the former has 50% of the instances with (0.4, 0) and 50% with (0.2, 0.2); the latter has 90% with (0.4, 0) and 10% with (0.2, 0.2). We can see that in both settings, EMI +op1+op2 outperformed the other two variants, further validating the effectiveness of the proposed optimization techniques.

Since EMI +op1+op2 achieved the best performance, we will only use it in TARS and compare it with the existing cleaning strategies.

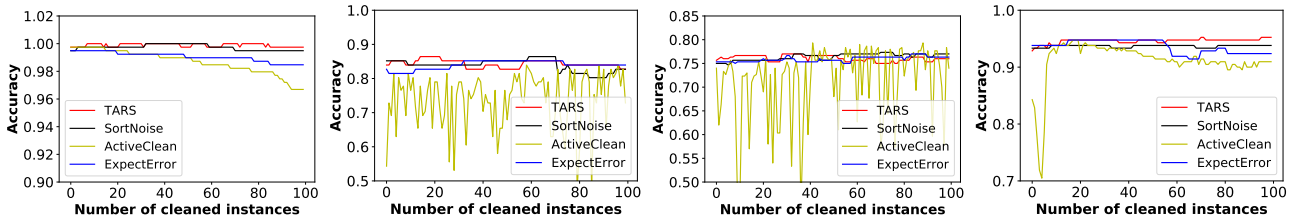
6.3.2 Cleaning Strategies. Like the previous experiments, we first set the noise rates to (0.4, 0), leading to a big gap between learning with noisy labels and with true labels. We trained a model on the data and then asked an oracle to clean the data. Figure 12 compares the model's accuracy w.r.t. different cleaning strategies on the Gaussian, Heart, German, and Cancer datasets for a cleaning budget of 100 instances.

We have two observations from the results. First, TARS outperformed SortNoise and ExpectError on all four datasets. The reason is that TARS considers both label noise and model changes in its cleaning strategy while SortNoise and ExpectError only considers one of them. Second, we find that the performance of ActiveClean may not be very stable for our problem setting (see Figure 12(b) and Figure 12(c)). This is because that ActiveClean is focused on a different cleaning scenario, where there is a large dirty dataset



(a) Gaussian (b) Heart (c) German (d) Cancer

Figure 12: Comparing the cleaning strategy of TARS with the existing cleaning strategies (Noise Rates: (0.4, 0)).



(a) Gaussian (b) Heart (c) German (d) Cancer

Figure 13: Comparing the cleaning strategy of TARS with the existing cleaning strategies (Noise Rates: (0.2, 0.2)).

(both features and labels are dirty), and an oracle cleans the data in batches. It is common that each batch contains at least 100 instances, but in this experiment, the entire cleaning budget was only 100.

The above experiment demonstrated that when the initial model's accuracy was much lower than the final model's accuracy, the cleaning strategies were able to keep improving the model's accuracy through cleaning. A natural question is that when the initial model is already good enough, will the cleaning strategies hurt the model's performance? To answer this question, we set the noise rates to (0.2, 0.2) such that the noisy labels did not impose a significant negative impact on the model's performance. Figure 13 shows the result. We can see that TARS did a good job for keeping the model's accuracy as the good as the initial model's accuracy.

Mix of Noise Rates. We also compared TARS with the existing cleaning strategies on the datasets with a mix of noise rates. The results can be found in Figures 16 and 17 in the Appendix. They further validated the observations described above.

7 RELATED WORK

Crowdsourcing. There are three research topics in crowdsourcing related to our work: task assignment [2, 4, 10, 21, 25, 40], truth inference [9, 13, 15, 27, 39], and active learning (from crowds) [14, 19, 22, 37]. Task assignment studies how to determine which task should be assigned to an incoming crowd worker. A number of task assignment algorithms have been proposed [2, 21, 40], but their objectives are not to maximize the performance of a supervised-learning model. Truth inference studies the problem of inferring the ground-truth label of each instance based on (inconsistent) labels from different workers. There are many interesting ideas proposed to solve this problem [13, 15, 27]. However, the goal of truth inference is to improve label quality rather than model quality. As discussed in Section 5.1, a good cleaning strategy should not only consider noise rates but also model changes. Active learning (from crowds) determines which unlabeled instance should be sent to a crowd worker to label. Since a crowd worker may make mistakes, there are some studies on the trade-off between asking another crowd worker to relabel an instance or label a new instance [19, 30]. In our problem, we consider that all the instances have been labeled by the crowd and an oracle can be used to clean the instances.

Data Cleaning. Algorithmic data cleaning approaches have been improving in quality, but still far from perfect [5]. In view of the challenge, human-guided data cleaning has recently attracted a lot of attention [1, 6, 7, 11, 16, 24, 33–36, 38]. The existing studies can be broadly divided into two categories. One category is to leverage humans (either crowd workers or experts) to solve a particular data-cleaning problem, such as entity resolution [7, 11, 33, 35], missing value imputation [24], and data repairing [6, 36]. The other category is to clean data for a particular data analysis task, such as building a machine-learning model [16] and answering SQL queries [1, 34]. Our work belongs to this category. In particular, we are focused on cleaning crowdsourced labels for supervised learning, which is a problem that has not been explored before.

Learning with Noisy Labels. There is a large body of work in the Machine Learning community on learning with noisy labels (see [12] for a survey). Some existing approaches aim to develop a robust algorithm to tolerate label noise [20, 31]. Liu et al. [20] use importance reweighting to ensure that any surrogate loss function can be used for classification, proving that the label noise involved in training will not affect the search for an optimal classifier. Sukhbaatar et al. [31] train neural networks on images with noisy labels directly, but add an extra layer designed to model the label noise. There are also some works [3, 32] that seek to leverage data cleaning for model training. Brodley and Friedl [3] use ensembles of classifiers of identify mislabeled instances and then remove them from the training data. Veit et al. [32] use a small sample of clean data to reduce the severity of label noise on the (much larger) noisy dataset. In contrast to these works, this paper is focused on a different data-cleaning scenario, i.e., oracle-based label cleaning.

8 CONCLUSION

In this paper, we have studied the problem of cleaning crowdsourced labels using oracles for supervised learning. We developed TARS, a label-cleaning advisor that can provide data scientists with two pieces of advice when they need to train or/and test a model using noisy labels. We formally defined the corresponding problems: model evaluation and cleaning strategy. For the first problem, we described effective techniques to estimate the model's true performance as well as bound the estimation error, for different evaluation

metrics (accuracy, precision, recall, F-score). For the second problem, we devised a new cleaning strategy, called EMI, to overcome the limitations of the existing cleaning strategies, and developed two techniques to further optimize its effectiveness. The experimental results show that (1) TARS can accurately estimate the model's true performance, with the estimation error up to 3× smaller than DirtyEval; (2) TARS can improve the model accuracy by a larger margin than ActiveClean, SortNoise, and ExpectError, for the same cleaning budget.

REFERENCES

[1] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1199–1214, 2015.

[2] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. C. Tan. Asking the right questions in crowd data sourcing. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 1261–1264, 2012.

[3] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *J. Artif. Intell. Res.*, 11:131–167, 1999.

[4] P. Cheng, X. Lian, L. Chen, J. Han, and J. Zhao. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, 28(8):2201–2215, 2016.

[5] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *SIGMOD*, pages 2201–2206, 2016.

[6] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *ACM SIGMOD*, pages 1247–1261, 2015.

[7] S. Das, P. S. G. C., A. Doan, J. F. Naughton, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, and Y. Park. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *ACM SIGMOD*, pages 1431–1446, 2017.

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

[9] X. L. Dong and F. Naumann. Data fusion - resolving data conflicts for integration. *PVLDB*, 2(2):1654–1655, 2009.

[10] J. Fan, G. Li, B. C. Ooi, K. Tan, and J. Feng. icrowd: An adaptive crowdsourcing framework. In *ACM SIGMOD*, pages 1015–1030, 2015.

[11] D. Firmani, B. Saha, and D. Srivastava. Online entity resolution using an oracle. *PVLDB*, 9(5):384–395, 2016.

[12] B. Frénay and M. Verleysen. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014.

[13] J. Gao, Q. Li, B. Zhao, W. Fan, and J. Han. Truth discovery and crowdsourcing aggregation: A unified perspective. *PVLDB*, 8(12):2048–2049, 2015.

[14] D. Haas, J. Wang, E. Wu, and M. J. Franklin. Clamshell: Speeding up crowds for low-latency data labeling. *PVLDB*, 9(4):372–383, 2015.

[15] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. Minimizing efforts in validating crowd answers. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 999–1014, 2015.

[16] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959, 2016.

[17] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 3–12, 1994.

[18] G. Li, J. Wang, Y. Zheng, and M. J. Franklin. Crowdsourced data management: A survey. *IEEE Trans. Knowl. Data Eng.*, 28(9):2296–2319, 2016.

[19] C. H. Lin, Mausam, and D. S. Weld. Re-active learning: Active learning with relabeling. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1845–1852, 2016.

[20] T. Liu and D. Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

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

[22] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, and S. Madden. Scaling up crowd-sourcing to very large datasets: A case for active learning. *PVLDB*, 8(2):125–136, 2014.

[23] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1196–1204. Curran Associates, Inc., 2013.

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

[25] J. Pilourdault, S. Amer-Yahia, D. Lee, and S. B. Roy. Motivation-aware task assignment in crowdsourcing. In *EDBT*, pages 246–257, 2017.

[26] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

[27] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. G. Parameswaran, and C. Ré. Slimfast: Guaranteed results for data fusion and source reliability. In *SIGMOD*, pages 1399–1414, 2017.

[28] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 441–448, 2001.

[29] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[30] V. S. Sheng, F. J. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 614–622, 2008.

[31] S. Sukhbaatar and R. Fergus. Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, 2(3):4, 2014.

[32] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie. Learning from noisy large-scale datasets with minimal supervision. *arXiv preprint arXiv:1701.01619*, 2017.

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

[34] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *ACM SIGMOD*, pages 469–480, 2014.

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

[36] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, 4(5):279–289, 2011.

[37] Y. Yan, R. Rosales, G. Fung, and J. G. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1161–1168, 2011.

[38] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu. Cleaning uncertain data with a noisy crowd. In *ICDE*, pages 6–17, 2015.

[39] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

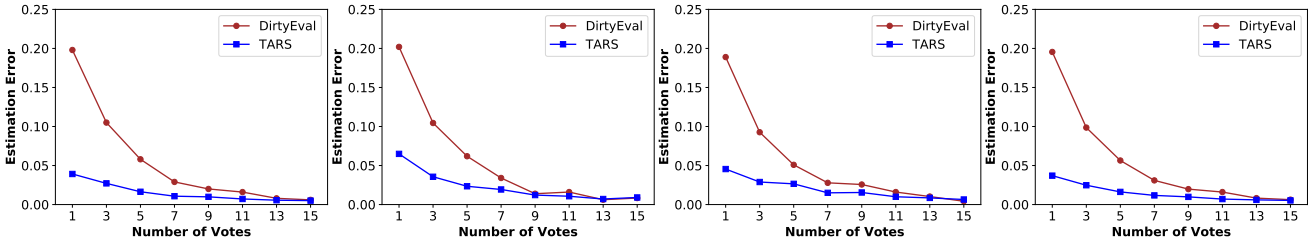
[40] Y. Zheng, J. Wang, G. Li, R. Cheng, and J. Feng. QASCA: A quality-aware task assignment system for crowdsourcing applications. In *ACM SIGMOD*, pages 1031–1046, 2015.

APPENDIX

A ADDITIONAL EXPERIMENTS

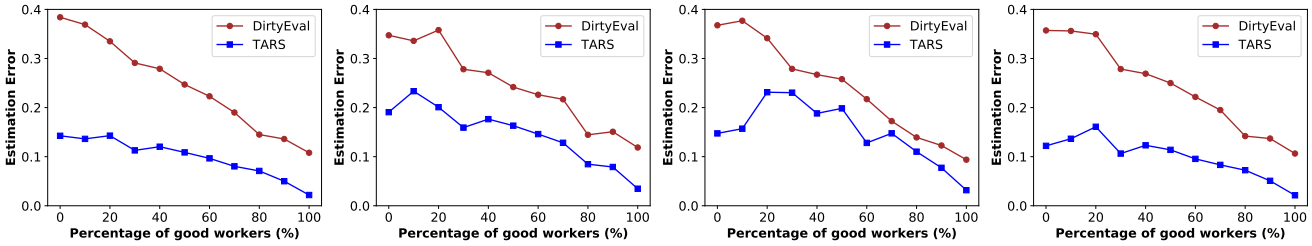
Number of Votes. In Section 6.2.1, we assume that each instance has a single vote (i.e., labeled by a single worker). Next, we relax the assumption and investigate how adding the number of votes will affect the estimation error. We varied the number of votes from 1 to 15. Figure 14 reports the comparison results between TARS and DirtyEval w.r.t. different evaluation metrics. We can see that increasing the number of votes reduced the estimation error exponentially. This is because that as the increase of the number of votes, the noise rate of a consolidated label will decrease exponentially. We also see that TARS outperformed DirtyEval for not only a single-vote situation but also multiple-vote situations. Eventually, their estimation error will both converge to zero.

Percentage of Good Workers. In a real-world crowdsourcing setting, there could be a mix of high-quality and low-quality workers. We designed an experiment aiming to examine whether TARS can still achieve good performance in this situation. We consider that a "good" worker has the noise rate of 0.9, and a "bad" worker has the noise rate of 0.4. We varied the percentage of the instances labeled by good workers. Figure 15 reports the result. We can see that TARS achieved much smaller estimation error than DirtyEval for all situations. For example, when 50% of the instances were labeled by good workers, TARS got an estimation error of about 0.1 (w.r.t. accuracy and F-score), while the estimation error of DirtyEval was more than 2× larger.



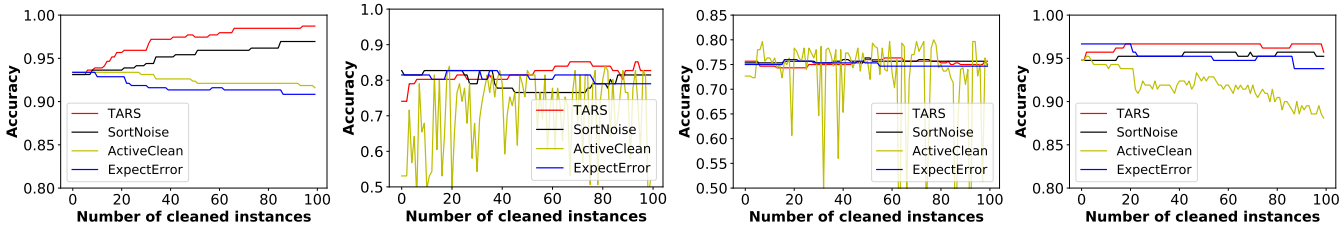
(a) accuracy (b) precision (c) recall (d) F-score

Figure 14: Comparison of the estimation error between TARS and DirtyEval by varying the number of votes per instance (Gaussian).



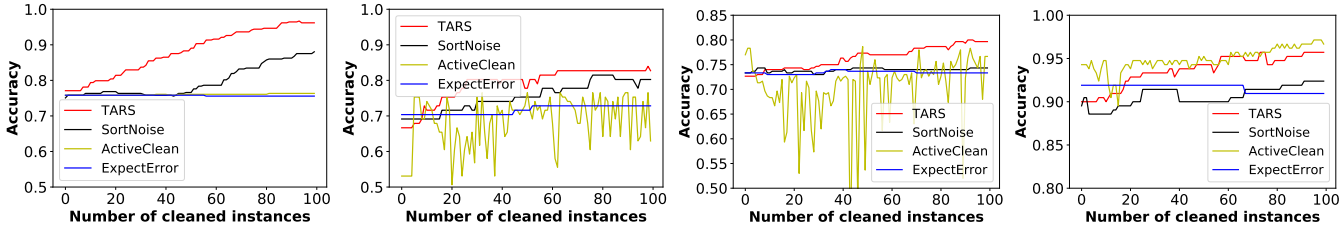
(a) accuracy (b) precision (c) recall (d) F-score

Figure 15: Comparison of the estimation error between TARS and DirtyEval by varying the percentage of good workers (Gaussian).



(a) Gaussian **(b) Heart** **(c) German** **(d) Cancer**

Figure 16: Comparing the cleaning strategy of TARS with the existing cleaning strategies (Noise Rates: 50% of (0.4, 0) and 50% of (0.2, 0.2)).



(a) Gaussian **(b) Heart** **(c) German** **(d) Cancer**

Figure 17: Comparing the cleaning strategy of TARS with the existing cleaning strategies (Noise Rates: 90% of (0.4, 0) and 10% of (0.2, 0.2)).