Cleaning Crowdsourced Labels Using Oracles for Statistical Classification

Mohamad Dolatshah†, Mathew Teoh†, Jiannan Wang†, Jian Pei◇†
Simon Fraser University† JD.com◇
{mdolatsh, mteoh, jnwang, jpei}@cs.sfu.ca

ABSTRACT

Nowadays, crowdsourcing is being widely used to collect training data for solving classification problems. However, crowdsourced labels are often noisy, and there is a performance gap between classification with noisy labels and classification with ground-truth 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 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. ground-truth labels. Secondly, in the model training stage, given a training dataset with noisy labels, and a classification 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 novel cleaning strategy along with two optimization techniques, and illustrate that it is superior to the existing cleaning strategies. We evaluate 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; and (2) TARS can improve the model accuracy by a larger margin than the existing cleaning strategies, for the same cleaning budget.

PVLDB Reference Format:
Mohamad Dolatshah, Mathew Teoh, Jiannan Wang, Jian Pei. Cleaning Crowdsourced Labels Using Oracles for Statistical Classification. *PVLDB*, 12(4): 376-389, 2018.
DOI: https://doi.org/10.14778/3297753.3297758

1. INTRODUCTION

Classification is a fundamental problem in machine learning and statistics. It has achieved great success in many real-world applications, such as entity resolution, disease prediction, and fraud detection. The goal of classification is to train a *classifier* (a.k.a. *model*) from a collection of ⟨instance, label⟩ pairs such that the classifier can predict the labels of unseen instances. The labeled pairs are 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.

Proceedings of the VLDB Endowment, Vol. 12, No. 4
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3297753.3297758

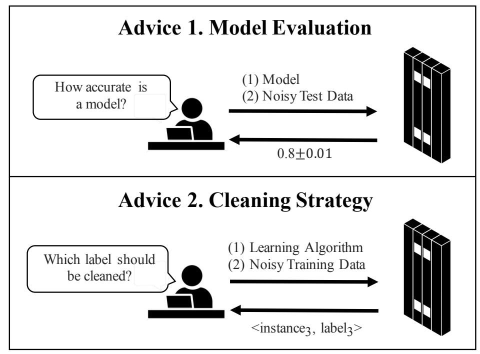


Figure 1: TARS can provide two pieces of valuable advice when data scientists need to train or test a model using noisy labels

Despite the great success that classification has achieved, it suffers from the high cost of data labeling. Crowdsourcing is a promising way to overcome this limitation. Crowdsourcing platforms such as Amazon Mechanical Turk have a large pool 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 ground-truth label using a truth-inference algorithm, the state-of-the-art truth-inference algorithms are still far from perfect [47].

Having noisy labels in training data will negatively affect the performance of a classification algorithm because the algorithm tries to predict noisy labels rather than ground-truth labels. In the Machine Learning community, there has already been some work that studies how to clean noisy labels to solve this problem [5]. 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 [15].

Unlike the existing work, our work draws some inspiration from the recent progress in the data cleaning community [3, 14, 20], 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 ground-truth label. This scenario more often than not holds in reality. Imagine a data scientist needs to train a good classifier 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 classification algorithm for noisy labels. Instead, we aim to develop a *label cleaning advisor*, named as TARS[1], that can

[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.

advise a data scientist on how to make the best use of oracle-based cleaning for classification. 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 current 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 the model will perform w.r.t. *ground-truth labels* (rather than w.r.t. noisy labels). Advice 1 is especially useful when cleaning a label is prohibitively expensive. Moreover, the model may sometimes need to be evaluated not only on a single test dataset but on multiple different ones. For example, suppose a data scientist wants to monitor the model's performance over time. Without Advice 1, she needs to keep asking domain experts to clean a new test dataset again and again, which is tedious and costly.
- *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. The size of a training dataset is typically in the range from a few hundred to tens of thousands [1, 2]. It is expensive to ask oracles to clean them all. With the help of Advice 2, data scientists are able to apply oracles to clean them *progressively*. For example, suppose there is a small cleaning budget (e.g., 50), but a training dataset is much bigger (e.g., 1000). In this situation, a data scientist can ask for Advice 2 iteratively for 50 times and identify which label should be cleaned at each iteration.

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 ground-truth label. We apply a given model to the test dataset and obtain the predicted label of each instance (see Table (c)). Since ground-truth labels are unknown, the model's true accuracy cannot be directly derived. One naive approach is to treat noisy labels as ground-truth 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 ground-truth 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 out 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 $\mathcal{O}(\frac{1}{\sqrt{N}})$, regardless of what β is, and (ii) as β decreases, the overall estimation error will decrease at a rate of $\mathcal{O}\big(\frac{1}{|0.5-\beta|}\big)$, 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. One straightforward solution is to apply active cleaning [36] (e.g., uncertain sampling) to determine which label should be cleaned. However, our problem setting is different since 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. In the end, we analyze the time complexity of our approach, and study how to improve its efficiency in practice.

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 classification. We identify two challenging problems (model evaluation and cleaning strategy) 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, and discuss how to make it run efficiently in practice.

- 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. ground-truth 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. PROBLEM DEFINITION

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

Classification With Ground-Truth 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 ground-truth label (see [11] for the extension to multiclass classification). Denote a sample drawn i.i.d. from $\mathbb{G}$ as $S = \{(x_i, y_i)\}_{i=1}^{N}$. A classification 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.

Crowdsourced Data. 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 $\mathcal{C} = \{(x_i, \mathcal{L}^i)\}_{i=1}^{N}$, which we call the *crowdsourced data*.

Worker Model. Typically, there are three ways to model worker quality. The first one is to use a single probability value, which represents the probability that a worker provides the correct label of an instance (regardless of its ground-truth label). The second one is to use a *confusion matrix*, which represents the probability that a worker provides the correct label of an instance given the ground-truth label. The third one is to use a probability distribution, which represents the probability that a worker provides the correct label of an instance given its difficulty. In this paper, we choose the second one because it has been widely used in the crowdsourcing literature and has been shown an effective way to model worker quality [23]. For the third approach, although it sounds more reasonable, the challenge is that in practice, it is hard to estimate how difficult a worker feels when doing a task. Without an accurate estimation, the worker model will not perform well.

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 ground-truth 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 ground-truth label y, worker w_j provides label l with probability of $q^{(j)}_{y,l}$.

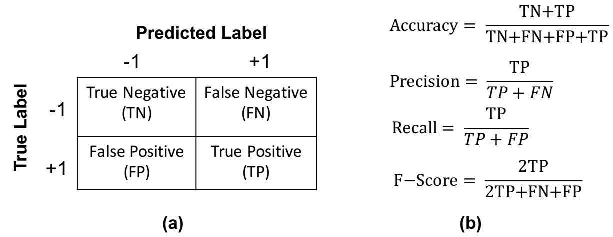


Figure 3: An illustration of (a) model's confusion matrix and (b) four evaluation metrics derived from the confusion matrix.

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 [23]. 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 [10] to iteratively estimate their values.

There are certainly many other approaches to improve the computation of workers' confusion matrices and the prior. In this paper, we treat this as an orthogonal problem, and assume that they are given as input of our problems.

Classification With Noisy Labels. Let $\mathcal{A}$ denote a classification algorithm that learns a classifier f on crowdsourced data $\mathcal{C}$. This problem has been well studied in the Machine Learning community [15, 32]. Their basic ideas are either to develop a robust classification 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 $\mathcal{C}$ 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 confusion matrix. We can see that it is similar to a worker's confusion matrix, where each row also represents a ground-truth 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)*, *False Positive (FP)*, *True Negative (TN)*, and *False Negative (FN)*. 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 $\mathcal{C}$, TARS aims to use $\mathcal{C}$ to estimate $\mathsf{eval}(\mathbb{G}, f)$. Let $\widetilde{\mathsf{eval}}(C, f)$ denote an estimation of $\mathsf{eval}(\mathbb{G}, f)$. We say $\widetilde{\mathsf{eval}}(C, f)$ to be unbiased if the expected value of the estimation is equal to the true value, i.e., $\mathbb{E}[\widetilde{\mathsf{eval}}(C, f)] = \mathsf{eval}(\mathbb{G}, f)$.

TARS computes a *confidence interval* to bound the estimation error of $\widetilde{\mathsf{eval}}(C, f)$. Suppose the estimated value is $\widetilde{\mathsf{eval}}(C, f) = 0.8$. 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 $\mathcal{C}$, a model f, and an evaluation metric* eval, TARS *aims to determine (1) an unbiased estimator, $\widetilde{\mathsf{eval}}(C, f)$, of the model's true performance, and (2) a confidence interval, $[\widetilde{\mathsf{eval}}(C, f) - \epsilon_1, \widetilde{\mathsf{eval}}(C, f) + \epsilon_2]$, 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 $\mathcal{C}$ 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 classification algorithm $\mathcal{A}$ takes crowdsourced data $\mathcal{C}$ as input and outputs a model f. By cleaning the labels in $\mathcal{C}$, 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 $\mathcal{C}$, a classification algorithm $\mathcal{A}$, and an evaluation metric* eval, *let f_i be the model resulting from cleaning instance x_i, then training on $\mathcal{C}$ with $\mathcal{A}$.* TARS *aims to determine which instance should be cleaned such that the model's true performance can be improved the most: $i^* = \operatorname*{argmax}_{i=1,\ldots,N} \mathsf{eval}(\mathbb{G}, f_i)$.*

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 ground-truth labels in $\mathbb{G}$. Rather than ground-truth labels, we only have worker labels. The trick to bridging the gap between worker labels and ground-truth labels is noticing that there are some relationships between them, which are captured by the workers' confusion matrices.

We use worker labels along with workers' confusion matrices to infer the most likely ground-truth label for each instance, and then figure out how to quantify the uncertainty of each inferred label. Specifically, given a crowdsourced dataset $\mathcal{C} = \{(x_i, \mathcal{L}^i)\}_{i=1}^N$, 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 ground-truth 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 [10] 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:

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

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

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

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 Quantify 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 ground-truth label, each column represents a consolidated label, and Each cell in the matrix is a conditional probability $r^{(\mathcal{W}_i)}_{y,y'} = P(Y_i' = y' \mid Y = y)$ $(y', y \in \{\pm 1\})$. If the context is clear, we abbreviate $r^{(\mathcal{W}_i)}$ as $r^{(i)}$ or r.

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

$$P(Y'_i \mid Y) = \sum_n P(Y'_i \mid \bar{\mathcal{L}}^i_n, Y)\, P(\bar{\mathcal{L}}^i_n \mid Y), \tag{3}$$

where $\{\bar{\mathcal{L}}^i_n : n = 1, 2, \cdots, 2^{k_i}\}$ 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: $\bar{\mathcal{L}}^i_1 = \{(w_1, -1), (w_2, -1)\}$, $\bar{\mathcal{L}}^i_2 = \{(w_1, -1), (w_2, +1)\}$, $\bar{\mathcal{L}}^i_3 = \{(w_1, +1), (w_2, -1)\}$, and $\bar{\mathcal{L}}^i_4 = \{(w_1, +1), (w_2, +1)\}$, where, e.g., $\bar{\mathcal{L}}^i_1$ means that w_1 provides -1 and w_2 provides -1.

The equation depends on two forms of conditional probabilities: $P(Y' \mid \bar{\mathcal{L}}^i_n, Y)$ and $P(\bar{\mathcal{L}}^i_n \mid Y)$. For the latter, we have already discussed how to compute it in Equation 23.

$$P(\bar{\mathcal{L}}^i_n \mid Y) = \prod_{(w_j,l)\in\bar{\mathcal{L}}^i_n} P(\bar{\mathcal{L}}^i_n = (w_j, l) \mid Y) = \prod_{(w_j,l)\in\bar{\mathcal{L}}^i_n} q^{(j)}_{y,l} \tag{4}$$

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

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

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

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. ground-truth labels) contributes to the bound (Section 4.2.1).

4.1 Estimating Model's True Performance

We first present our unified estimation framework. Recall that a model's performance (e.g., $\mathsf{accuracy}$, $\mathsf{F\text{-}score}$) 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 [28] 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 ground-truth label y. When ground-truth labels are accessible, we can represent TP as:

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

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 ground-truth 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. [28] 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 ground-truth 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 $\mathsf{accuracy}$: $\frac{\widetilde{\mathsf{TP}}+\widetilde{\mathsf{TN}}}{N}$. Similarly, we can get the estimated values for $\mathsf{precision}$: $\frac{\widetilde{\mathsf{TP}}}{\widetilde{\mathsf{TP}}+\widetilde{\mathsf{FP}}}$, recall: $\frac{\widetilde{\mathsf{TP}}}{\widetilde{\mathsf{TP}}+\widetilde{\mathsf{FN}}}$, and $\mathsf{F\text{-}score}$: $\frac{2\widetilde{\mathsf{TP}}}{2\widetilde{\mathsf{TP}}+\widetilde{\mathsf{FP}}+\widetilde{\mathsf{FN}}}$.

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

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

For $\mathsf{precision}$, recall, and $\mathsf{F\text{-}score}$, their denominator is not a constant value but a random variable. We do *not* have $\mathbb{E}[\frac{X}{Y}] = \frac{\mathbb{E}[X]}{\mathbb{E}[Y]}$. This type of estimator is called conditionally unbiased given Y. If Y is not fixed, there will be a bias. We did a literature review to explore how existing studies handle this bias, and found that many works adopt the same approach as us, i.e., approximate $\mathbb{E}[\frac{X}{Y}]$ as $\frac{\mathbb{E}[X]}{\mathbb{E}[Y]}$ [21,29,34,38]. Furthermore, there is a good theoretical guarantee on the quality of the approximation: $\mathbb{E}[\frac{X}{Y}] = \frac{\mathbb{E}[X]}{\mathbb{E}[Y]} + O(n^{-1})$ for $n \to \infty$, which shows that the approximation error becomes smaller and smaller as n increases [29]. We evaluate the effectiveness of the estimators of $\mathsf{precision}$, recall, and $\mathsf{F\text{-}score}$ in the experiments, and find that they perform well on both synthetic and real-world datasets.

Remark. Note that the unbiased loss estimator was originally proposed to solve a different problem (i.e., model training with noisy labels). It has never been used to solve our problem (i.e., model evaluation over crowdsourced labels). We are the first to study how to evaluate model performance using crowdsourced noisy labels.

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. Ground-Truth Labels. The other source of error comes from noisy labels. Since ground-truth 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. ground-truth labels.

To address this challenge, we develop an analytical confidence interval based on the Central Limit Theorem. 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.

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:

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

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

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

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 29, we find

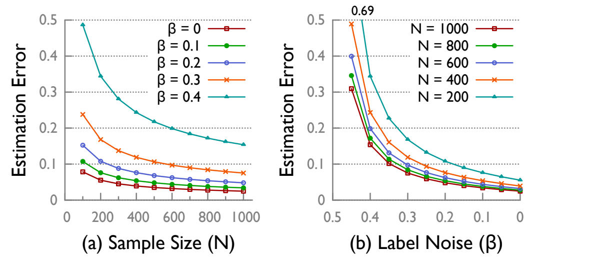


Figure 4: The relationships between sample size (N), label noise (β), and estimation error (half the width of the 95% confidence interval), for model accuracy $\theta = 0.8$. (For simplicity, we do not show the estimation error for $\beta \in (0.5, 1]$ because based on Equation 15, it will be the same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

that $\widetilde{\mathsf{Loss}}_{0/1}(t, y')$ can only take two possible values:

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

Suppose that $\widetilde{\mathsf{Loss}}_{0/1}(t, y')$ 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 $\widetilde{\mathsf{Loss}}_{0/1}(t, y')$ is:

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

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

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

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

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

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 model's true accuracy, 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 $\mathcal{O}\Big(\frac{1}{\sqrt{N}}\Big)$. For example, when sample size is increased from $N = 100$ to 1000, estimation error will decrease by about $\mathcal{O}\Big(\sqrt{\frac{1000}{100}}\Big) = 3$ times.

Insight 2. The second source of error (noisy label vs. ground-truth 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

(a) Before cleaning (b) After cleaning x_i

Figure 5: Illustrating the limitation of sorting by noise rate.

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

4.2.2 Empirical Confidence Interval

Now we present how to use the bootstrap to compute empirical confidence intervals for other estimators than accuracy. 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. Given D and an estimator $\widetilde{\mathsf{eval}}(D, f)$, 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, $\widetilde{\mathsf{eval}}(D^{(1)}, f)$, $\widetilde{\mathsf{eval}}(D^{(2)}, f)$, $\cdots$, $\widetilde{\mathsf{eval}}(D^{(n)}, f)$. Given a confidence level (e.g., 95%), let $\widetilde{\mathsf{eval}}_{2.5\%}$ and $\widetilde{\mathsf{eval}}_{97.5\%}$ denote the 2.5th and 97.5th percentile of the distribution, respectively. Then, the 95% confidence interval is denoted by $[\widetilde{\mathsf{eval}}_{2.5\%}, \widetilde{\mathsf{eval}}_{97.5\%}]$.

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

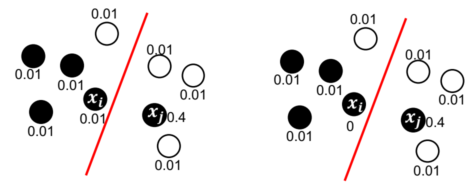
Now we present how TARS provides the second piece of advice: which label should be cleaned? Please note that, unlike the previous section, here we turn our focus to the *model training* stage. In the following, we first explain why the existing cleaning strategies do not work in Section 5.1, and then 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 provided the reasons and propose effective solutions in Section 5.3.

5.1 Limitations of Existing 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. 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. 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 (the red line). However, x_i's label only has a noise rate of 0.01, which is very unlikely to flip after cleaning. 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.

Sorting by Noise Rate. Another cleaning strategy is to 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



(a) Before cleaning (b) After cleaning x_i

Figure 6: Illustrating the limitation of uncertain sampling.

after cleaning, but it totally ignores the impact to the model. Consider an example in Figure 5. This strategy will select x_i for cleaning because it has the greatest noise. As shown in Figure 5(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.

ActiveClean. 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 [20]. ActiveClean predicts the ground-truth label of each instance, and then estimates how cleaning each instance would change the model based on predicated ground-truth 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 ground-truth 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.

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:

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

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

$$i^* = \underset{i}{\operatorname{argmax}} \ \mathsf{EMI}(i) \qquad (17)$$

In this paper, we are focused on oracle-based cleaning. That is, we only use oracles to clean instances. An interesting observation is that the data-dependency relationship could also be leveraged to clean instances. For example, consider an instance x_i. If an oracle confirms that x_i has a positive (or negative) label, it is very likely that its neighbors (i.e., the instances that are close to x_i based on some distance function) also have positive (or negative) labels. We discuss this extension in the technical report [11].

Next, we discuss how to compute $\mathsf{EMI}(i)$, which has 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 ground-truth 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

$$P(\mathsf{case}_2) = \frac{P(Y'_i = y'_i | Y_i = -y'_i) \cdot P(Y_i = -y'_i)}{P(Y'_i = y'_i)} \propto P(Y'_i = y'_i | Y_i = -y'_i) \cdot P(Y_i = -y'_i) \quad (18)$$

$P(Y'_i = y'_i | Y_i = -y'_i)$ 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

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

Similarly, we can obtain

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

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

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

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 $P(\mathsf{case}_2)$, EMI tends to not select x_i. Consider x_i in Figure 5. Since the model would not change after flipping the label of x_i, leading to a zero value of $\mathsf{eval}(\mathbb{G}, f_i) - \mathsf{eval}(\mathbb{G}, f)$, EMI will not select x_i.

Time Complexity. Let $N = |\mathcal{D}|$ denote the size of training data. EMI needs to retrain N models for each iteration. Let $cost_{\mathsf{train}}$ denote the average cost of training a model on $\mathcal{D}$. The cost of each iteration is $\mathcal{O}(N \cdot cost_{\mathsf{train}})$. Suppose there are t iterations. Then, the total cost of our approach will be $\mathcal{O}(t \cdot N \cdot cost_{\mathsf{train}})$. Based on our observation of real-world datasets [1, 2], the training data size N is typically in the range from a few hundred to tens of thousands. Without improving the efficiency, it is expensive to apply our approach to a training dataset with thousands of instances.

In practice, there are two simple but effective ideas that can be used to improve the efficiency.

(1) *Pruning*. Rather than retrain N models for every iteration, we can prune $p\%$ of the iterations, where $p\%$ is a parameter to balance the trade-off between efficiency and effectiveness. For example, 50% means that we prune 50% of the iterations and retrain models for every other iteration. This pruning idea works well in practice because if $\mathsf{EMI}(i)$ is high in one iteration, since the noise rate keeps unchanged, its value tends to be high in later iterations. As a result, we can use the value of $\mathsf{EMI}(i)$ from a previous iteration to approximate its value in a later iteration. For the above example, if $p\% = 90\%$, then the cost will be reduced by $10\times$.

(2) *Parallel Computing*. At each iteration, we can retrain N models in parallel. Recall that these models are trained on $\mathcal{D}_1, \mathcal{D}_1, \cdots, \mathcal{D}_N$, where $\mathcal{D}_i$ only has one instance different from $\mathcal{D}$ for all $i \in [1, N]$. We keep one copy of training data $\mathcal{D}$ in memory. When training a model on $\mathcal{D}_i$, we read $\mathcal{D}$ and flip the label of the i-th instance on-the-fly. In this way, the training data $\mathcal{D}$ is read-only, and there is no coordination when training multiple models in parallel. For the above example, suppose CPU has 16 cores. The total cost will be reduced by $16\times$. In addition to parallel computing, we can leverage distributed computing frameworks (e.g., Spark) or GPU computing to further reduce the running time. 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 $\mathsf{eval}(\mathbb{G}, f_i)$ and $\mathsf{eval}(\mathbb{G}, f)$. One natural idea is to use $\mathcal{D}_i$ for $\mathsf{eval}(\mathbb{G}, f_i)$ and $\mathcal{D}$ for $\mathsf{eval}(\mathbb{G}, f)$ because f_i is trained on $\mathcal{D}_i$ and f is trained on $\mathcal{D}$:

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

However, there are two issues about this idea.

First, as shown in Equation 28, the goal is to estimate the difference between $\mathsf{eval}(\mathbb{G}, f_i)$ and $\mathsf{eval}(\mathbb{G}, f)$ as more accurate as possible. Let X and Y denote the estimators of $\mathsf{eval}(\mathbb{G}, f_i)$ and $\mathsf{eval}(\mathbb{G}, f)$, respectively. That is, we aim to minimize $\mathsf{var}(X - Y) = \mathsf{var}(X) + \mathsf{var}(Y) - \mathsf{cov}(X, Y)$. In order to minimize $\mathsf{var}(X - Y)$, we need to increase $\mathsf{cov}(X, Y)$ 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 $\mathcal{D}_{\mathsf{train}}$ and a validation dataset $\mathcal{D}_{\mathsf{vdn}}$. Only the instances in $\mathcal{D}_{\mathsf{train}}$ can be cleaned and be used to train a model; the instances in $\mathcal{D}_{\mathsf{vdn}}$ cannot be cleaned or train a model, and their job is to estimate $\mathsf{eval}(\mathbb{G}, f_i)$ and $\mathsf{eval}(\mathbb{G}, f)$:

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

It is worth noting 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 $\mathsf{EMI}(i)$ for different instances i can be very similar. To see why this is the case, consider a simple situation where $\mathsf{eval}(\mathbb{G}, f)$ 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 $P(\mathsf{case}_2)$.

This means if we have two instances i_1 and i_2, for which $\mathsf{eval}(\mathbb{G}, f_{i_1}) - \mathsf{eval}(\mathbb{G}, f)$ and $\mathsf{eval}(\mathbb{G}, f_{i_2}) - \mathsf{eval}(\mathbb{G}, f)$ are very

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 labels)

Dataset	Size	Positive%	#Dimension	Type
Gaussian	1000	50%	2	SS
Gaussian (big)	10000	50%	2	SS
Heart	270	44%	13	RS
Cancer	699	31%	9	RS
Diabetes	768	65%	20	RS
Restaurant	2004	5%	8	RR

similar, the resulting values $\mathsf{EMI}(i_1)$ and $\mathsf{EMI}(i_2)$ will be very similar. In the worst case, $\mathsf{EMI}(i_1) = \mathsf{EMI}(i_2)$, which makes it impossible to distinguish which instance would be a better candidate for cleaning.

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

Before multiplying by $u(x_i)$, we need to transform $\mathsf{EMI}(i)$ into a positive value, using some function $\sigma\colon \mathbb{R} \to \mathbb{R}^+$. In order to preserve the relative ordering of $\mathsf{EMI}(i)$, 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:

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

Thus, with this optimization, we choose instance i^* to clean by computing: $i^* = \underset{i}{\operatorname{argmax}} \ \mathsf{MU}(i)$.

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 $\sigma_{x_2} = \sigma_{y_2} = 1$. Gaussian (big) is generated using the same process but the size is ten times larger. (2) Heart, Diabetes, and Cancer are three real-world datasets downloaded from the UCI Machine Learning Repository[2]. 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 [41]. Table 1 illustrates the detailed statistical information of the six data sets.

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

Noisy Labels. The noisy labels for the Gaussian, Heart, Diabetes, 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 ground-truth label is -1, then it will be flipped with a probability of 0.2; if its ground-truth 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 crowd 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 $r_{+1,-1} \in [0, 0.2]$.

Baselines for Advice 1. We compared TARS with two existing model evaluation approaches.

- DirtyEval treats noisy labels as ground-truth labels and directly evaluates the model accuracy over the data.
- NoiseRemove [5] uses a set of learning algorithms to create classifiers to identify mislabeled instances. It removes the instances that are identified as mislabeled and evaluates the model accuracy over the remaining data.

Baselines for Advice 2. We compared TARS with six existing cleaning strategies.

- Random cleans the instance randomly selected from the uncleaned instances.
- SortNoise cleans the instance whose noisy label is most likely to be wrong, without considering the impact on the current model.
- ActiveClean cleans the instance which, if the instance was cleaned, would impart the greatest change to the current model.
- Uncertainty Sampling (USample) [22] cleans the instance that the current model is most uncertain about, without considering the noise rate.
- Expected Error Reduction (ExpectError) [35] cleans the instance such that the current model's error can be reduced the most. This is similar (in spirit) to our strategy. But the error is computed on noisy labels rather than estimated w.r.t. ground-truth labels.
- Hung [19] is a state-of-the-art truth-inference approach. Unlike other approaches, it allows an oracle to validate crowd answers. However, its strategy is to clean the instance in order to maximize label quality rather than model quality.

To examine the benefit of oracle-based cleaning, we also compared with two non-oracle-based cleaning algorithms.

- NoiseRemove trains a classifier on the data with noisy labels filtered by [5].
- NoiseTolerate [28] trains a classifier using a loss function tolerant to label noise.

The code was written in Python 2.7. We trained logistic regression models on all datasets using scikit-learn[3]. Each dataset was randomly divided into a training set and a test set with the ratio of 2 to 1. The experiments were run on a Windows machine with an Intel Core 8 i7-6700 3.40GHz processor and 16GB of RAM.

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 data.

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

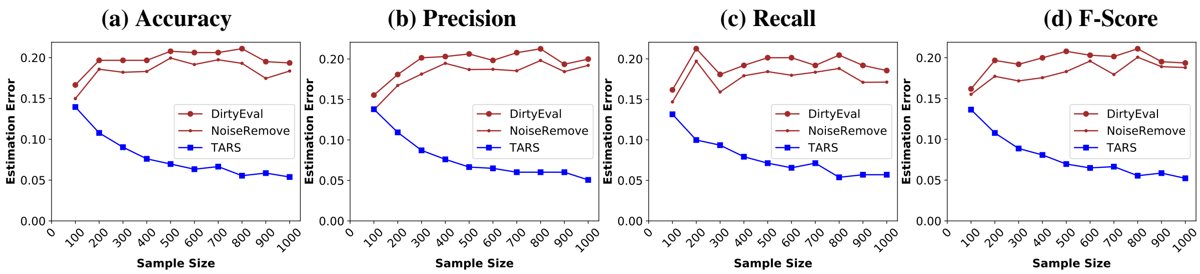


Figure 7: Comparison of different approaches for Advice 1 by varying sample size (Gaussian).

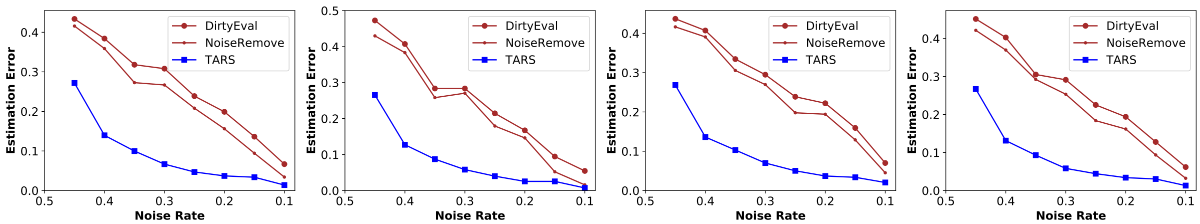


Figure 8: Comparison of different approaches for Advice 1 by varying noise rates (Gaussian).

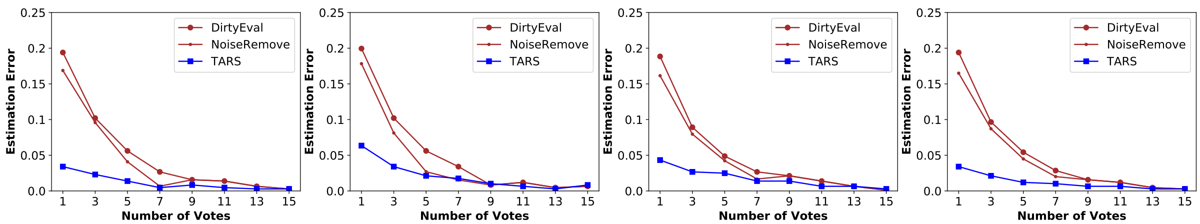


Figure 9: Comparison of different approaches for Advice 1 by varying the number of votes per instance (Gaussian).

6.2.1 *Sensitivity Analysis*

We evaluate the estimation error of TARS on the Gaussian dataset by varying the sample size, the noise rate, and the number of votes, 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, and the number of votes is 1. 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 and NoiseRemove.

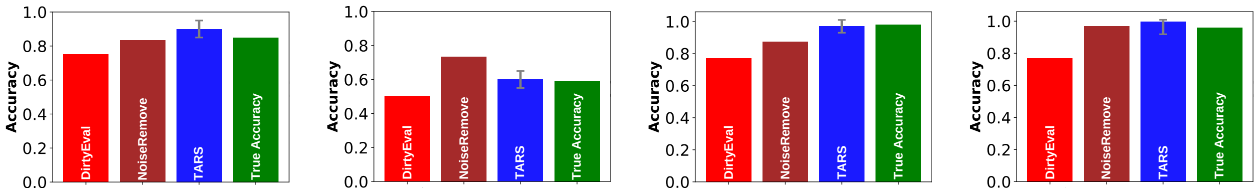
Sample Size. Figure 7 compares the estimation error of TARS, DirtyEval, and NoiseRemove by varying the dataset size from 100 to 1000, w.r.t. different evaluation metrics. We can see that as the sample size was increased by 10 times, the estimation error of TARS was reduced by about 3 times, but DirtyEval and NoiseRemove did not change so much. The reason is that TARS takes into account noise rates, which converts label error (i.e., bias) into sampling error (i.e., variance). Increasing the sample size can reduce the sampling error at the rate of $\mathcal{O}(\frac{1}{\sqrt{N}})$. In comparison, DirtyEval has a much larger label error but with a smaller sampling error. While the sampling error can still be decreased, since the overall estimation error was dominated by the label error, the improvement was marginal. NoiseRemove is able to remove some mislabeled instances (i.e., reduce label error), but still does not touch many of them in the dataset. This is because that the classifiers that NoiseRemove uses to identify mislabeled instances are trained over *noisy* data rather than ground-truth data. If these classifiers give a wrong prediction to a mislabeled instance, NoiseRemove fails to remove this instance from the data.

Noise Rate. Figure 8 compares the estimation error of TARS, DirtyEval, and NoiseRemove 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 the rate of $\mathcal{O}\big(\frac{1}{|0.5-\beta|}\big)$. This experiment validated that it holds for the other evaluation metrics empirically.

Number of Votes. We investigate how adding the number of votes will affect the estimation error. We varied the number of votes from 1 to 15. Figure 8 reports the result. 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 and NoiseRemove for not only a single-vote situation but also multiple-vote situations. Eventually, their estimation error will both converge to zero.

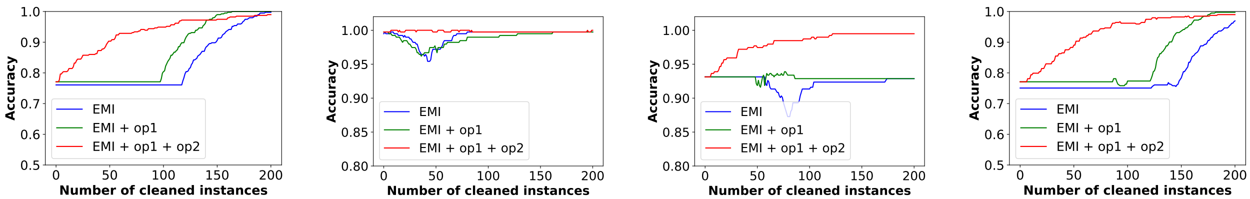
6.2.2 *Performance on Real-world Datasets*

We evaluate TARS on the four real-world datasets. We used the noise rate of 0.2 for Heart, Diabetes, and Cancer, and used the real-world crowd workers to label Restaurant. We tested all evaluation metrics, but due to space constraints, we can only show some of them in the paper. We chose Accuracy for Heart, Diabetes, and Cancer since they have balanced labels, and Accuracy is the most widely used evaluation metric in this situation. However, Accuracy is not suitable for the Restaurant dataset since the dataset has very unbalanced labels (positive# : negative# = 1:24). Simply labeling everything as negative will lead to an accuracy of $\frac{24}{25} = 96\%$. Therefore, we chose F-score for the Restaurant dataset. 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 Diabetes and Cancer datasets, TARS returned almost the same accuracy as the true accuracy while DirtyEval and NoiseRemove performed much worse. On the Heart and Restaurant datasets, TARS and NoiseRemove had similar per-



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

Figure 10: Comparison of different approaches for Advice 1 on real-world datasets.



(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).

formance. But please note that unlike TARS, NoiseRemove does not provide a confidence interval for its estimated result. If a data scientist does not know how far an estimated result is from the true result, she will not be able to trust the estimated result even if it is close to the true result.

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.

6.3.1 Optimization Techniques

In Section 5.3, we identified the possible issues when applying EMI in practice, and proposed two optimization techniques, denoted by op1 and op2, 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 various 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.

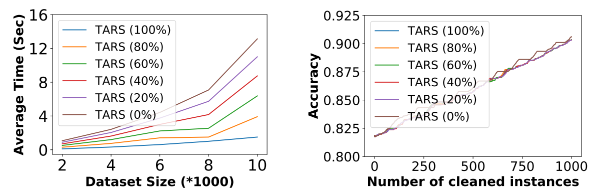
In Figure 11(b), we set the noise rates to $r_{-1,+1} = 0.2$ and $r_{+1,-1} = 0.2$. In this setting, we can see that the model's accuracy was (almost) not affected by the noisy labels. Therefore, all three cleaning strategies started with a very accurate model. After sending some instances to an oracle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

Note that op1 and op2 are heuristic approaches, thus they are not guaranteed to always improve the performance of EMI. Nevertheless, Figure 11 empirically shows that they help to improve the performance in most situations. Since EMI +op1+op2 typically performs the best, we will only use it in later experiments.

6.3.2 Efficiency Study

We proposed two techniques, parallel computing and pruning, in Section 5.2 to optimize the efficiency of our cleaning strategy. The result for the use of parallel computing is obvious, which will



(a) Efficiency **(b) Effectiveness**

Figure 12: Evaluating the impact of the pruning technique on the efficiency and effectiveness of TARS.

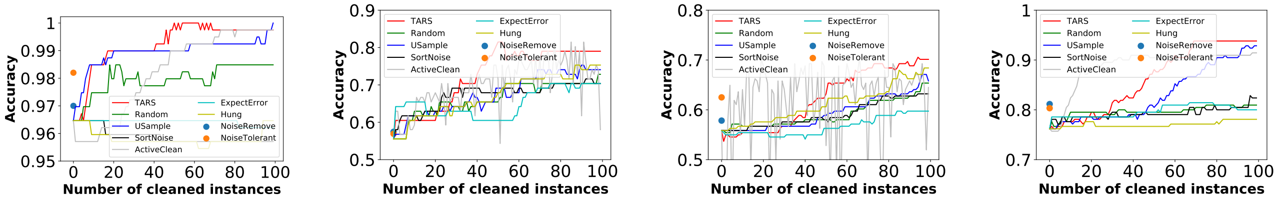
improve the efficiency by c times without hurting the effectiveness, where c is the number of CPU cores. So, we only show the results for the pruning technique. We used the Gaussian (big) dataset and set the noise rates to (0.4, 0). Let TARS$(p\%)$ denote our cleaning strategy with the pruning percentage of p%. For example, TARS(0%) means that no pruning is adopted; TARS(50%) means that 50% of the iterations are pruned; TARS(100%) means that all the iterations (except for the first one) are pruned.

We varied the data size from 2000 to 10,000, and calculated the average time that each approach spent over 1000 iterations. Figure 12(a) shows the result. We can see that the pruning technique can significantly improve the efficiency. For example, without pruning, TARS(0%) spent about 13 seconds in deciding which instance to clean; with 80% pruning, TARS(0%) reduced the average time to less than 4 seconds. Next, we evaluate the impact of the pruning technique on the effectiveness of TARS. Figure 12(b) shows the result. We can see that using pruning only had a slightly negative impact on the accuracy. This is because that when data is large, cleaning a few instances will not change the current model a lot, thus there is no need to retrain all the models for every iteration.

6.3.3 Cleaning Strategies

In this section, we first compare oracle-based cleaning with non-oracle-based cleaning, and then evaluate different cleaning strategies for oracle-based cleaning.

Oracle-based vs. Non-Oracle-based Cleaning. Our paper is focused on oracle-based cleaning. NoiseRemove and NoiseTolerate represent non-oracle-based cleaning. To motivate the need for oracle-based cleaning, we compared NoiseRemove and NoiseTolerate with TARS by varying the number of instances cleaned by an oracle. Figure 13 shows the results. NoiseRemove and NoiseTolerate are both represented by a single dot in the figure since they do not use oracles to clean noisy labels. We have two observations from the figures. First, there is a performance gap between NoiseRemove (or NoiseTolerate) and learning with ground-truth labels in terms of classification accuracy. Second, TARS is able to fill the gap by



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

Figure 13: Comparing the effectiveness of different cleaning strategies by varying the number of cleaned instances.

progressively asking an oracle to clean noisy labels. These results motivate the need to study oracle-based cleaning.

Oracle-based Cleaning Strategies. We varied the number of cleaned instances, and compared the model's accuracy of different cleaning strategies. We constructed a mix of noise rates: $(0.4, 0.1)$ and $(0.3, 0.4)$, and applied the first noise rate to two thirds of the instances, and applied the second one to the remaining instances.

Figure 13 shows the results. We have three interesting observations. First, TARS outperformed all the other cleaning strategies. The reason is that TARS considers both label noise and model changes in its cleaning strategy while Random does not consider either of them, SortNoise and Hung do not consider model changes, and USample and ExpectError do not consider label noise. Second, the performance of ActiveClean sometimes was not very stable (see Figure 13(b) and Figure 13(c)). This is because that ActiveClean is focused on a different cleaning scenario, where there is a large dirty dataset (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. Third, after a number of instances cleaned, the performance of TARS becomes stable. This is because that TARS prioritizes cleaning those instances likely to affect the model. If such instances have all been cleaned, the model will become stable.

Section 5.3 presents two optimization techniques (op1 and op2) for TARS. One natural question is whether these optimizations are also applicable to the other cleaning strategies. Let us first take a look at op1. op1 divides noisy data into two disjoint parts, where one is used to train a model and the other is used to test the model. Among all the other cleaning strategies, op1 can only be applied to ExpectError. In the above experiment, we have already implemented op1 in ExpectError. For op2, it uses model uncertainty to break the tie. This is a general technique that can be applied to all cleaning strategies. We evaluated the impact of op2 on each cleaning strategy, and got some interesting observations. First, op2 has little impact on USample because USample has already taken into account model uncertainty. Second, the performance of ActiveClean is still not stable even with op2. Third, for the other cleaning strategies, op2 can sometimes bring some improvements, but TARS still performs the best since it is the only approach that considers both label noise and model impact.

7. RELATED WORK

Crowdsourcing. There are three research topics in crowdsourcing related to our work: task assignment [4, 6, 13, 17, 26, 31, 48], truth inference [12, 16, 19, 33, 47], and active learning (from crowds) [18, 24, 27, 45]. Task assignment studies how to determine which task should be assigned to an incoming crowd worker. However, their objectives are not to maximize the performance of a classification model. Truth inference studies the problem of inferring the ground-truth label of each instance based on (inconsistent) labels from different workers. For traditional truth-inference techniques where no oracle-based cleaning is involved [47], this is orthogonal to our problem because we can first use them to get a noisy label along with noise rate for each instance, and then apply TARS to train or test a model using the noisy labels. The oracle-based cleaning was adopted by a recent truth-inference study [19]. However, as shown in the experiment, TARS outperformed this approach because it aims to maximize label quality rather than model quality. 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 [24, 37]. 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 [7]. In view of the challenge, human-guided data cleaning has recently attracted a lot of attention [3, 8, 9, 14, 20, 30, 41, 42, 43, 44, 46]. 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 [9, 14, 41, 43], missing value imputation [30], and data repairing [8, 44]. The other category is to clean data for a particular data analysis task, such as building a machine-learning model [20] and answering SQL queries [3, 42]. Our work belongs to the second category. In particular, we are focused on cleaning crowdsourced labels for statistical classification, 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 [15] for a survey). Some existing approaches aim to develop a robust algorithm to tolerate label noise [25, 39]. There are also some works [5, 40] that seek to leverage data cleaning for model training. 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 statistical classification. We developed TARS, a label-cleaning advisor that can provide data scientists with two pieces of advice when they need to train or 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. We developed two techniques to further optimize its effectiveness and proposed two ideas to improve its efficiency. The experimental results show that (1) TARS can accurately estimate the model's true performance, with the estimation error up to $3\times$ smaller than DirtyEval and NoiseRemove; (2) TARS can improve the model accuracy by a larger margin than the state-of-the-art cleaning strategies, for the same cleaning budget.

Acknowledgements. This work was supported in part by the NSERC Discovery Grant program, the Canada Research Chair program, the NSERC Strategic Grant program, the NSERC CRD Grant program, the NSERC Engage Grant program, and an SFU President's Research Start-up Grant. All opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

9. REFERENCES

[1] Crowdflower: Data For Everyone. `https://www.figure-eight.com/data-for-everyone/`. Accessed: 2018-05-29.

[2] UCI Machine Learning Repository: Classification Datasets. `https://archive.ics.uci.edu/ml/datasets.html?format=&task=cla`. Accessed: 2018-05-29.

[3] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *ACM SIGMOD*, pages 1199–1214, 2015.

[4] R. Boim, O. Greenshpan, T. Milo, S. Novgorodov, N. Polyzotis, and W. C. Tan. Asking the right questions in crowd data sourcing. In *ICDE*, pages 1261–1264, 2012.

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

[6] 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.

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

[8] 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.

[9] 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.

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

[11] M. Dolatshah, M. Teoh, J. Wang, and J. Pei. Cleaning crowdsourced labels using oracles for statistical classification. Technical Report, 2018.

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

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

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

[15] 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.

[16] 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.

[17] J. Gao, X. Liu, B. C. Ooi, H. Wang, and G. Chen. An online cost sensitive decision-making method in crowdsourcing systems. In *ACM SIGMOD*, pages 217–228, 2013.

[18] 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.

[19] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. Minimizing efforts in validating crowd answers. In *ACM SIGMOD*, pages 999–1014, 2015.

[20] 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.

[21] D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In *SIGIR*, pages 246–254. ACM, 1995.

[22] D. D. Lewis and J. Catlett. Heterogenous uncertainty sampling for supervised learning. In *ICML*, pages 148–156, 1994.

[23] 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.

[24] C. H. Lin, Mausam, and D. S. Weld. Re-active learning: Active learning with relabeling. In *AAAI*, pages 1845–1852, 2016.

[25] 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.

[26] 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.

[27] 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.

[28] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari. Learning with noisy labels. In *NIPS*, pages 1196–1204, 2013.

[29] S. Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *CVPR*, pages 548–555, 2014.

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

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

[32] 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.

[33] 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.

[34] S. H. Rice. A stochastic version of the price equation reveals the interplay of deterministic and stochastic processes in evolution. *BMC evolutionary biology*, 8(1):262, 2008.

[35] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *ICML*, pages 441–448, 2001.

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

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

[38] C. G. Small. *Expansions and asymptotics for statistics*. Chapman and Hall/CRC, 2010.

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

[40] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. J. Belongie. Learning from noisy large-scale datasets with minimal supervision. In *IEEE CVPR*, pages 6575–6583, 2017.

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

[42] 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.

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

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

[45] Y. Yan, R. Rosales, G. Fung, and J. G. Dy. Active learning from crowds. In *ICML*, pages 1161–1168, 2011.

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

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

[48] 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. MULTI CLASS GENERALIZATION

In this section, we talk about how we can generalize our approach to the multi-class classification problem.

Classification With True Labels. In order to be able to handle multi-class classification, we have to support all class labels for y. Therefore, our definition would be $y = l_k$ where $l_k \in Y = \{l_1, ..., l_n\}$ representing the set of all possible true labels.

Crowdsourced Data. Corresponding to the true labels we also have to change the definition for noisy labels. Let $\mathcal{L}^i = \{(w_j, l_{i,j}) \mid w_j \text{ labels } x_i\}$ denote the corresponding labels, where $l_{i,j} \in Y$ represents the label given by worker w_j.

Worker Model. The confusion matrix of each worker w_j is a $n \times n$ matrix, where n represents the number of classes. For example, for 3-class classification we have:

$$q^{(j)} = \begin{bmatrix} q^{(j)}_{l_1,l_1}, & q^{(j)}_{l_1,l_2}, & q^{(j)}_{l_1,l_3} \\ q^{(j)}_{l_2,l_1}, & q^{(j)}_{l_2,l_2}, & q^{(j)}_{l_2,l_3} \\ q^{(j)}_{l_3,l_1}, & q^{(j)}_{l_3,l_2}, & q^{(j)}_{l_3,l_3} \end{bmatrix}$$

where each row represents a true label, each column represents a worker's provided label, and $q^{(j)}_{y,l}$ $(y \in Y, l \in Y)$ means that given an instance with true label y, worker w_j provides label l with probability of $q^{(j)}_{y,l}$.

A.1 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 n conditional probabilities: $P(Y_i = l_k \mid \mathcal{L}^i)$, the probability that instance x_i has a true label of l_k, conditioned on the labels that the workers provide. We will choose whatever label in this set that has the biggest conditional probability.

$$\begin{aligned} P(Y_i = l_k \mid \mathcal{L}^i) &= \frac{P(\mathcal{L}^i \mid Y_i = l_k)P(Y_i = l_k)}{P(\mathcal{L}^i)} \\ &\propto P(\mathcal{L}^i \mid Y_i = l_k)P(Y_i = l_k) \\ &= P(Y_i = l_k) \prod_{l_{i,j} \in \mathcal{L}^i} P(L_j = l_{i,j} \mid Y_i = l_k) \\ &= P(Y = l_k) \prod_{l_{i,j} \in \mathcal{L}^i} q^{(j)}_{l_k, l_{i,j}} \end{aligned} \quad (23)$$

A.2 Estimating Model's True Performance

For estimating model's performance we have to generalize the proposed unbiased estimator to handle multiple classes. The way to do it is to consider the loss function for each of the possible classes:

$$\mathsf{Loss}(t, l_k) = \sum_n P(Y' = l_i \mid Y = l_k) \cdot \widetilde{\mathsf{Loss}}(t, l_i) \quad (24)$$

Therefore we will have n equations and n unknown variables. By solving these n equations we can have the estimated loss function for each class $\widetilde{\mathsf{Loss}}(t, l_i)$.

A.3 Expected Model Improvement

As discussed before we have the notion of improvement signal as follow:

$$\mathsf{EMI}(i)_k = P(\mathsf{case}_k) \cdot \Big(\mathsf{eval}(\mathbb{G}, f_i) - \mathsf{eval}(\mathbb{G}, f)\Big) \quad (25)$$

Then we can generalize this concept for each of the class labels:

$$\begin{aligned} P(\mathsf{case}_k) &= \frac{P(Y'_i = y'_i | Y_i = l_k) \cdot P(Y_i = l_k)}{P(Y'_i = y'_i)} \\ &\propto P(Y'_i = y'_i | Y_i = l_k) \cdot P(Y_i = l_k) \end{aligned} \quad (26)$$

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

$$P(\mathsf{case}_k) \propto r^{(i)}_{l_k, y'_i} \cdot P(Y = l_k) \quad (27)$$

We can get the improvement signal for all of the possible classes and then for each record we have the final improvement signal as the maximum improvement signal among class label.

$$\mathsf{EMI}(i) = \max \mathsf{EMI}(i)_k \quad (28)$$

$$\widetilde{\mathsf{acc}} = \sum_{i=1}^{N} \frac{(1-\beta) \cdot I(t = y') - \beta \cdot I(t = -y')}{1 - 2\beta}. \quad (29)$$

A.4 Empirical Experiment

We evaluated our method on multi class Thyroid Disease dataset from UCI archive containing 3 classes and 215 instances. For simulating error in real data, noise rate of (0.1,0.1,0.3) corresponding to each class, was applied to the dataset. We varied the number of cleaned instances, and compared the models accuracy w.r.t. different cleaning strategies on this dataset. The result is shown in Figure 14.

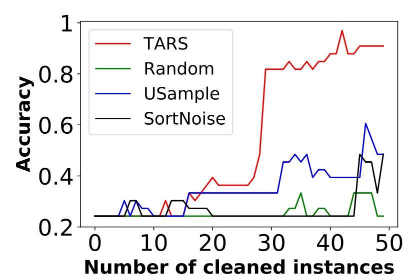


Figure 14: Comparison of cleaning strategies on the multi-class dataset

B. LEVERAGE DATA-DEPENDENCY RELATIONSHIPS

In this paper, we are focused on oracle-based cleaning. That is, we can only use oracles to clean instances. An interesting observation is that the data-dependency relationship could also be leveraged to clean instances. For example, consider an instance x_i. If an oracle confirms that x_i has a positive (or negative) label, it is very likely that its neighbors (i.e., the instances that are close to x_i based on some distance function) also have positive (or negative) labels.

Based on the above idea, we can extend Equation 16 as follows. Suppose a user specifies a distance function and a threshold k. In the following, when we say *cleaning an instance*, we will not only

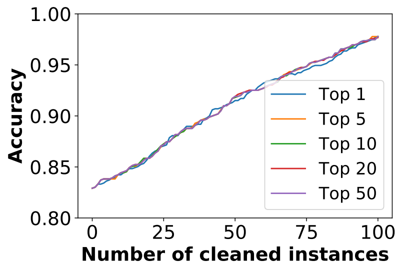


Figure 15: Evaluating the top-k approach that allows data scientists to clean k instances at a time.

use an oracle to clean the instance itself but also leverage the data-dependency relationship to clean its k neighbors. Let f denote the current model. Let f_i^1 denote the model trained after cleaning instance x_i, where the cleaned label is the *same* as the original noisy label. Let f_i^2 denote the model trained after cleaning instance x_i, where the cleaned label is *different* from the original noisy label. Then, the expected model improvement is defined as

$$\begin{aligned}\mathsf{EMI}(i) =& P(\mathsf{case}_1) \cdot (\mathsf{eval}(\mathbb{G}, f_i^1) - \mathsf{eval}(\mathbb{G}, f)) \\ &+ P(\mathsf{case}_2) \cdot (\mathsf{eval}(\mathbb{G}, f_i^2) - \mathsf{eval}(\mathbb{G}, f))\end{aligned}$$

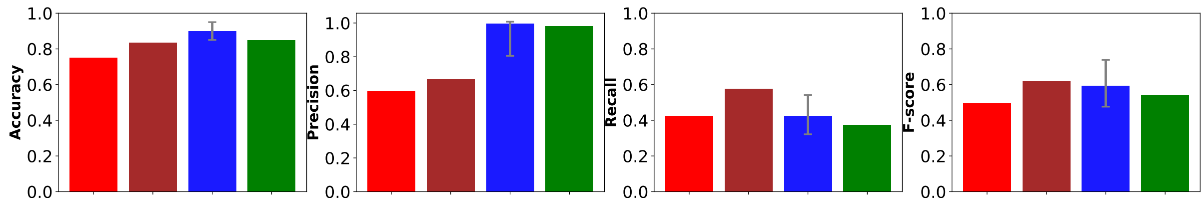
We can use the method (see Equations 18-21) described in the paper to compute $P(\mathsf{case}_2)$, and then get $P(\mathsf{case}_1) = 1 - P(\mathsf{case}_2)$. By plugging $P(\mathsf{case}_1)$ and $P(\mathsf{case}_2)$ into the above equation, we obtain $\mathsf{EMI}(i)$.

C. CLEANING K INSTANCES AT A TIME

In some situations, a data scientist may want to ask oracles to clean k instances at a time ($k > 1$). A simple solution is to compute the cleaning benefits of all instances using Equation 22, and then send the top-k instances (rather than the top-1 instance) to oracles to clean. We varied k from 1 to 50, and then compared the accuracy of TARS in different settings on the Gaussian dataset. Figure 15 shows the result. We can see that there is not much difference for different k values in terms of model accuracy. Thus, we can use this idea to extend TARS to clean more than one instances at a time.

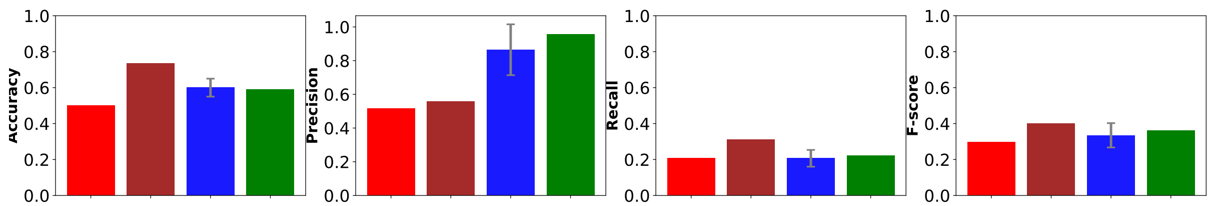
D. OTHER EVALUATION METRICS

Figures 16-19 show the complete result of Figure 10 (i.e., all four evaluation metrics over all four datasets).



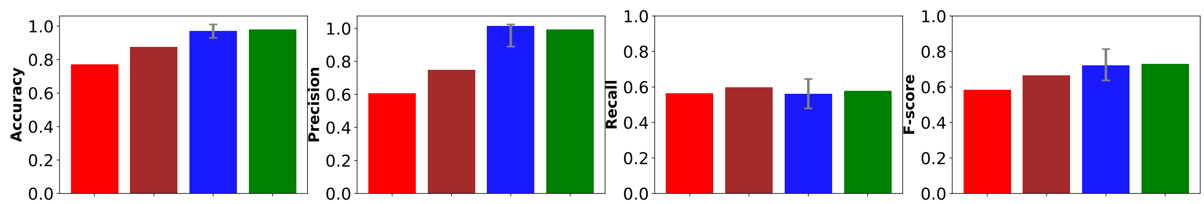
(a) Accuracy (b) Precision (c) Recall (d) F-Score

Figure 16: Comparison of different approaches for Advice 1 (Heart).



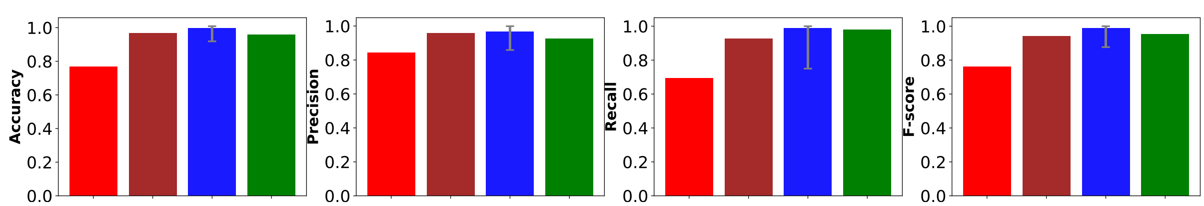
(a) Accuracy (b) Precision (c) Recall (d) F-Score

Figure 17: Comparison of different approaches for Advice 1 (Diabetes).



(a) Accuracy (b) Precision (c) Recall (d) F-Score

Figure 18: Comparison of different approaches for Advice 1 (Cancer).



(a) Accuracy (b) Precision (c) Recall (d) F-Score

Figure 19: Comparison of different approaches for Advice 1 (Restaurant).