Detecting and Preventing Confused Labels in Crowdsourced Data

Evgeny Krivosheev
University of Trento, Italy
evgeny.krivosheev@unitn.it

Siarhei Bykau*
Airbnb, USA
siarhei.bykau@airbnb.com

Fabio Casati
Servicenow, USA
fabio.casati@servicenow.com

Sunil Prabhakar
Purdue University, USA
sunil@purdue.edu

ABSTRACT

Crowdsourcing is a challenging activity for many reasons, from task design to workers' training, identification of low-quality annotators, and many more. A particularly subtle form of error is due to *confusion* of observations, that is, crowd workers (including diligent ones) that confuse items of a class i with items of a class j, either because they are similar or because the task description has failed to explain the differences.

In this paper we show that confusion of observations can be a frequent occurrence in many tasks, and that such confusions cause a significant loss in accuracy. As a consequence, confusion detection is of primary importance for crowdsourced data labeling and classification. To address this problem we introduce an algorithm for confusion detection that leverages an inference procedure based on Markov Chain Monte Carlo (MCMC) sampling. We evaluate the algorithm via both synthetic datasets and crowdsourcing experiments and show that it has high accuracy in confusion detection (up to 99%). We experimentally show that quality is significantly improved without sacrificing efficiency. Finally, we show that detecting confusion is important as it can alert task designers early in the crowdsourcing process and lead designers to modify the task or add specific training and information to reduce the occurrence of workers' confusion. We show that even simple modifications, such as alerting workers of the risk of confusion, can improve performance significantly.

PVLDB Reference Format:
Evgeny Krivosheev, Siarhei Bykau, Fabio Casati, and Sunil Prabhakar. Detecting and Preventing Confused Labels in Crowdsourced Data. *PVLDB*, 13(11): 2522-2535, 2020.
DOI: https://doi.org/10.14778/3407790.3407842

1. INTRODUCTION

Crowdsourcing is often used today to both solve AI-like problems such as classification as well as to generate labeled training data for Machine Learning (ML) algorithms. Crowd labeled data may and most often will be biased or noisy [34], and abundant literature has proposed methods for addressing such challenges, from task design methodologies, to ways of improving or coping with poor task design [7, 29, 13, 47], worker testing strategies (to exclude low-quality annotators), and aggregation of annotations by several workers. For example, *truth finder* methods have been recently proposed [14, 27] to identify the correct annotation based on dependencies between workers [10], heterogeneity of values [25], data correlation [33], or different types of workers' accuracy and similarity [46, 23, 12]. Similarly, literature in crowdsourcing has proposed a number of algorithms to collect and aggregate crowd votes with the goal of minimizing errors in annotations, from simple Majority Voting to variations of Expectation Maximization [28, 11, 43], to algorithms that focus specific classes of problems, such as *screening problems* (identifying items that satisfy a conjunction of predicates) [31, 21, 22].

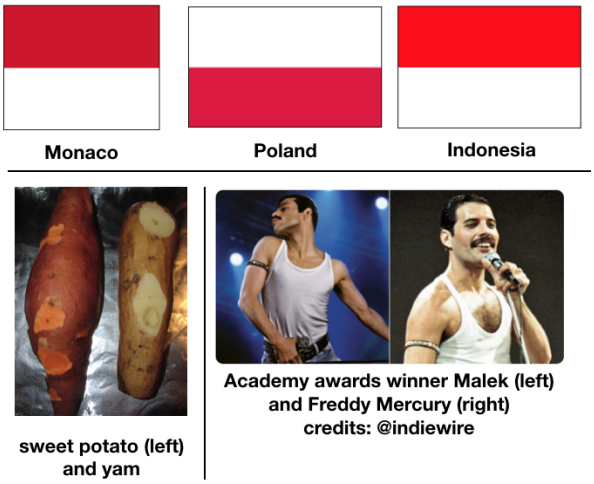

Figure 1: Examples of common confusions in flags and food as well as actors - in this case intentionally trying to be similar to a person.

A particularly subtle form of annotation errors is due to *confusion* of observations, that is, crowd workers (including diligent ones) that mistake items of a class i for items of a class j (for example, confuse pictures of actresses Ellen Barkin and Cameron Diaz, or Pancakes with Russian *Blini*)

*This work was done while the author was at Purdue University.

Proceedings of the VLDB Endowment, Vol. 13, No. 11
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3407790.3407842

often because they are similar or because the task description has failed to explain the differences. This problem can lead to a subset of workers consistently making errors, even on tasks that are apparently easy. For example, in a Google image search for "Monaco flag", 3 of the top 6 results (50%!) are wrong - and indeed, confused with the flag of Indonesia, and, to a lesser degree, Poland. We get similar error rates in many cases (e.g., Yams and Sweet potatoes, see Figure 1), pointing to the fact that for some classes of objects it is very likely to get erroneous ground truth labels, and that sources one could consider reputable (such as flags shop) contribute to the confusion. As we will show, *confusion can also occur in cases where the overall workers accuracy is very high* and therefore likely to go unnoticed since it is not a problem that surfaces across the board, but for some classes only. If left undetected and uncorrected, this leads to *consistently wrong labeling* for some classes, including the most popular and widely used ones, from majority voting (MV) to Dawid-Skene (D&S).

In this paper we define the problem and propose an algorithm for detecting such confusions. *Our primary goal is to detect confusion early in the crowdsourcing process* (that is, after a small number of votes), so that we can alert the task designers and improve the task. Our *secondary goal* is to be able to process crowdsourced labels to detect and *correct confused labels*. Once we do this we can then apply our favorite vote aggregation method to get class labels, from MV to D&S to the many interesting variations of Expectation Maximization proposed in the literature.

Specifically, we approach the problem by i) explicitly modeling confusion of observations as random variables, separate from workers' accuracy, and ii) by introducing the notion of clusters of confused items, i.e., groups of items which are likely to be confused by the crowd. In other words, we separate errors due to confusion (which have to do with items being similar) from errors due to low accuracy of workers (which may be due to many different reasons), and this allows us to better deal with datasets that have many items at risk for confusion. Then, we propose a novel inference algorithm based on Markov Chain Monte Carlo.

We evaluate the proposed algorithm over both synthetic datasets and crowdsourcing experiments, aiming at collecting datasets of different natures and difficulties. We show how the proposed inference procedure can be integrated with the main state-of-the-art aggregation techniques, that it scales both in the number of items and in the number of crowd workers, and that it significantly outperforms commonly used aggregation methods especially when the number of votes per item is relatively small (in the single digit range), as is common in many crowdsourcing tasks. We also analyze in which cases strong baselines such as D&S become competitive even in the presence of confused labels. We show that the accuracy of the proposed algorithm in *detecting* confusions is high (up to 99%), and truth discovery performance that results from *correcting* confusion errors is significantly improved. We propose a simple but effective automated method for modifying the crowdsourcing task once confusion is detected to reduce the probability of confusion occurring again as the crowdsourcing task proceeds.

2. RELATED WORK

The problem of *truth discovery*, i.e., integrating data from sources which provide conflicting data, has been extensively studied in the last decade. A number of approaches which model the accuracy of sources and probability of facts, termed *truth finders*, have been proposed [27]. Truth finders take the information about observed item labels and output the accuracy of sources as well as the probability of item labels being true. There are several methods truth finders employ to do so, such as link based analysis [19, 6] (to identify authoritative sources), optimization based methods [5, 24] (that is, formulate the truth discovery as an optimization problem), and finally, probabilistic graphical models [32, 41] (based on variables and their dependencies). Moreover, modern truth finders also take into account various factors such as dependency between sources [10], heterogeneity of values [25], data correlation [33], different types of source accuracy [46] and so on.

Truth discovery with confusion errors is a challenging problem both from the efficiency and effectiveness standpoints. First, we need to extend the existing truth finders with the ability to model and reason about possible confusion errors. No existing truth finder solution is capable of doing that and it is not trivial because we need to specify all probabilistic dependencies between sources and items. Second, to detect and take into account confusion errors we need to test all possible confusions and see which of them are likely to exist. That is a computationally expensive operation which requires a search over an exponential number of possibilities (there are 2^{N_c} states for N_c possible confused observations).

Crowdsourced classification has also been studied within the Human Computation community, following the seminal work of Dawid and Skene [8]. These studies cover a diversity of probabilistic data models, estimate the labeling procedure in the presence of items of different difficulties [43, 44], consider the confidence of workers' answers [39], apply EM-based algorithms to labeling in the presence of confusion matrix [28], discover truth of spatial events in mobile crowdsourcing [36]. A survey [18] investigates the models and algorithms on truth discovery inference in crowdsourcing as well as summarizes existing datasets and available tools [36]. An orthogonal thread of research focused on reducing noise in the first place. In this context, guidelines and practices for implementation and task design have been proposed for high-quality label collection via crowd and AI collaboration [3, 4].

The previous work in truth discovery falls short of finding and repairing such confusions. A typical truth finder or crowd classification algorithm would treat a confused labeling as workers' errors and lower their accuracy estimate in general, as opposed to recognizing that workers are generally accurate but on specific sets of labels. To the best of our knowledge only few papers deal with the problem, besides the above-mentioned Dawid-Skene on which we come back in the experiments section as it is part of our baselines. Liu and colleagues [28] study confusions and propose an efficient way to derive a confusion matrix for each worker. Their approach is tailored to situations where the number of workers is small (they focus on highly trained and skilled workers who rate a large number of items and on cases where the item-worker matrix is dense, testing the approach with three workers). Giancola and colleagues [14] also focus on workers' confusion matrix and propose the use of worker *style* matrices for every worker aiming at efficiently aggregating labels that can be systematically confused by a worker in text clustering and image segmentation tasks. The approach

is focused on performing permutation-invariant inference of class labels, in the sense that it is robust to worker-specific class permutations (e.g., different workers assigning different names to the class labels in a clustering problem). Our problem is different in that we do not focus on permutations, but on identifying confusions in labeling where class labels are given and (from the perspective of the task designer) clear, such as labeling picture with flags or food or names of famous actors, based on whatever the task requires. We also do not aim at learning parameters for workers' confusions (that is, at learning parameters for each worker) as our interest is instead in *groups of classes* that can be confused. Therefore the number and types of parameters of our model that have to be learned from data are chosen both to do this effectively and to be able to scale in the number of workers, items, and labels.

The topic of confusion is also gaining popularity in the context of machine learning, to identify and diagnose weaknesses of a ML model which often occurs in the presence of easily "confusable" classes and in general of noisy training data. For example, [17] propose a method to detect classes with high probability of confusion based on the output of an image classification ConvNet. The authors observe that the problem often arises due to a combination of weaknesses of the classifier and errors in the training data. Indeed, errors in the training dataset are known to significantly hamper the performance of the trained model [38], and the risk of noisy crowdsourced labels increases with the complexity of the problem and with the granularity of the classes [40]. The solution proposed is essentially based on identifying cliques of classes such that, for many items, classes in the cliques are among the most likely in the opinion of the ML model. Our problem is different as in crowdsourcing we have a crowd of votes (without explicit confidence information) rather than one voter (the ML model) that provides a distribution of class probabilities for each item.

3. MODEL AND APPROACH

3.1 Modeling Confusions

We model the problem as a classification task that takes in input a set (I, S, L) where $I = \{i_1, \ldots, i_m\}$ is the set of items to classify, S is the set of sources of information (in crowdsourcing, these are crowd workers), and $L = \{L_1, .., L_m\}$ is the set of all possible labels. Each item i have a number of possible labels (classes) we can assign $L_i = \{l_i^1, \ldots, l_i^{k_i}\}$ where k_i is the total number of distinct labels about i. Table 1 contains a summary of the notations used in this paper.

The votes of workers are denoted as $\Psi = \{\psi_{s,l}^i\}$, where $\psi_{s,l}^i = 1$ if source s labeled item i with label l, and $\psi_{s,l}^i = 0$ otherwise. Therefore, once we collect the votes we have a dataset $D = \langle I, S, \Psi, L \rangle$.

We extend this basic model by explicitly modeling confusions. Specifically, we introduce the notion of *clusters*, that group items that are at possible risk of confusion because they are "similar". As we will see, clusters act like *hypothesis* of confusions that inform the confusion detection algorithm, which will then focus on inferring if there is confusion among items in a cluster as well as which votes are likely "confused", and correct them. As we will see, clusters

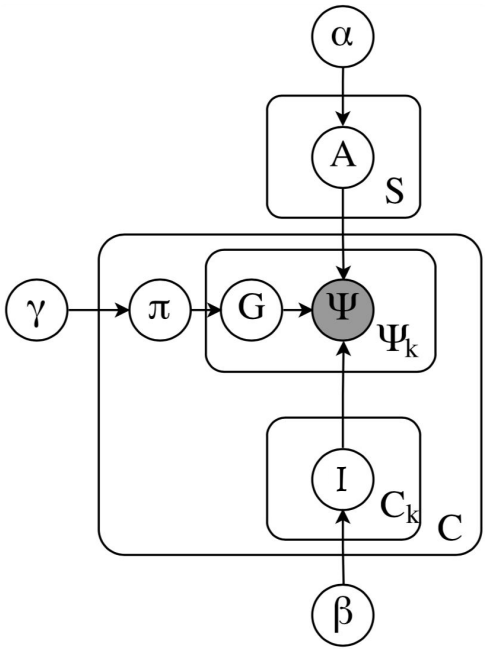

Figure 2: The probabilistic graphical model of Latent Truth Finder (MCMC-C).

Table 1: The summary of notations used in the paper.

Sign	Specification
$S = \{s_1, \ldots, s_n\}$	set of workers (sources)
$I = \{i_1, \ldots, i_m\}$	set of items
$L = \{L_1, \ldots, L_m\}$	all observed labels in data
L_i	labels observed only on item i
l_i^*	true label for item i
$\Psi = \{\psi_{s,l}^i\}$	set of all votes
Ψ_k	observations on the items in C_k
Ψ^s	observations of worker s
$\hat{\Psi}^i$	observations on item i including confused ones
$C = \{C_1, \ldots, C_{n_k}\}$	set of all clusters
C_k^{-i}	set of items in C_k except i
A_s	accuracy of worker s
G_k	confusions of cluster C_k
$G_{i,s}$	confusion of observation ψ_s^i
$G_k^{-i,s}$	confusions in cluster k except $G_{i,s}$
π_k	probability of confusion in C_k
α, β, γ	hyper-parameter

therefore help the algorithm in separating *confusion* errors (which can occur even in attentive, competent and dedicated workers) from other errors, and lead to a more effective estimation of workers' accuracy.

If task designers are aware of possible confusions they can formulate such hypotheses (define clusters) manually or via rules. For example, we can state that pictures of "Monaco" and "Indonesia" (Figure 1) are in the same cluster (manual definition), or that, in crowdsourcing tasks about labeling points of interest (i.e, reporting whether co-located restaurants are open or closed), neighboring points (restaurants in the same building) are to be put in the same cluster (rule-based definition). However, we expect that in many cases

cluster identification may be onerous or difficult, and in the experiment section we show how clustering methods can be applied to allow a fully automated confusion detection process. Regardless of the approach (manual, rule-based or automated), items with a risk of confusion are split into a set of mutually exclusive clusters, denoted by $C = \{C_1, \ldots, C_{n_k}\}$ where $C_k = \{i, j\}$ is a group of items i and j whose labels we believe can be confused.

3.2 Truth Finder with Confused Observations

We base our work on truth finder approaches. A truth finder $\mathcal{F}$ is a function that takes the dataset D as input, and outputs a set A of accuracy estimations (one per worker) and the class probability distribution P for each item:

$$\mathcal{F} : D \rightarrow \langle P, A \rangle,$$

where for each item $i \in I$, $P(l_i^* = l) \in [0, 1]$ is the probability that item i has label l, where l_i^* is the ground truth label on i. The probabilities of the distinct values of i sum up to 1.

The typical truth finder goal is to infer the unobserved variables (the true labels) given the observable variables (the crowd votes). Truth finders are typically based on the Expectation-Maximization (EM) algorithm [15, 46]. EM iteratively computes the accuracy of a source given the correctness of the values it provides, and then computes the correctness of a value given the accuracies of the sources that support it [32, 46, 33, 41, 30, 26]. We extend the basic Bayesian truth finder with the notion of clusters and define its generative model based on the plate model shown in Figure 2. In the figure, nodes indicate random variables, parameters, and hyper-parameters, and dark-shaded nodes denote observations made by workers. Specifically, I represents the true value of an item, G represents whether a worker made a confusion on a particular item or not, i. e., if a worker confused item i with item j during the measurement of value. G, I are latent (or non observed) variables; A, π are parameters that represent accuracy of a worker and a probability to make a confusion in a given cluster for an average worker. Finally, α, β, γ are hyper-parameters that represent our belief about corresponding distributions. The directed edges show dependencies between the variables, e.g., A depends on α and so on. To simplify the presentation, we limit to the case where clusters are pairs of items where labels can be confused, though the same approach can be extended to larger clusters.

This is how the generative process works: for each worker s we draw its accuracy from a Beta distribution:

$$A_s \sim Beta(\alpha_0 + N_+^s, \alpha_1 + N_-^s),$$

where α_0 and α_1 represent our prior belief about the accuracy of workers. N_+^s can be seen as the number of times a worker gave the correct label, whereas N_-^s is the number of incorrect labels.

For each cluster $C_k \in C$, we draw its probability π_k of confusion within the cluster from a Beta distribution:

$$\pi_k \sim Beta(\gamma_0 + N_{conf}^k, \gamma_1 + \overline{N_{conf}^k}),$$

where γ_0 and γ_1 represent our prior knowledge about confusions in C_k. N_{conf}^k is the count of how many times workers confused labels within that given cluster, and $\overline{N_{conf}^k}$ is the number of not confused votes in the cluster.

For each crowd vote ψ_s^i in cluster C_k, i.e., $\psi_s^i \in \Psi_k$ and $\psi_s^i \in L_i$, we draw a confusion binary label, denoted by $G_{i,s}$, from a Bernoulli distribution with prior π_k:

$$G_{i,s} \sim Bernoulli(\pi_k),$$

where $G_{i,s}$ is 1 if worker s confused label on item i, and 0 otherwise. Notice that in the model we do not consider the *direction* of the confusion, that is, whether in cluster C_k the confusion is symmetric or not symmetric, for example, workers confuse item i with j but not j with i. As we will see, once confusion is detected by the inference procedure, then identifying direction can be easily done.

Finally, we draw a true label l_i^* for an item i from a multinomial distribution with β_i Dirichlet prior:

$$l_i^* \sim multi(\beta_i),$$

where β_i is a $|L_i|$-dimensional parameter which specifies the multinomial prior for item i.

4. INFERENCE

Based on the generative model presented in Section 3 we propose an inference algorithm to detects confused observations. Once we have detected them, we can *correct* them (the algorithm helps in finding the probabilities of true labels) or *prevent* them, by modifying the crowdsourcing task as discussed later. Our inference task is *maximum a posteriori* (MAP) where we look for an assignment of model parameters and hidden variables that maximize the likelihood of the observed data. Given our model, the likelihood is defined as follows (we denote it with LK to distinguish it from labels):

$$LK(A, \pi : \Psi) = \prod_{C_k \in C} \sum_{I,G} \prod_{\psi_s^i \in \Psi_k} P(\psi_s^i \mid A, \pi, I, G; \alpha, \beta, \gamma) \tag{1}$$

In the above formula, for each cluster C_k we compute a product of the probabilities of observed data (Ψ_k) marginalized by the hidden variables, I and G. Note that we will omit the hyper parameters α, β and γ in the following formulas.

Our task is to find $\hat{A}$ and $\hat{\pi}$ such that the likelihood function is maximized: $\hat{A}, \hat{\pi} = \underset{A,\pi}{argmax}\ LK(A, \pi : \Psi)$. Once we found $\hat{A}$ and $\hat{\pi}$ we can easily compute the values of hidden variables, i.e., I and G. We propose to employ an approximation inference procedure based on Markov Chain Monte Carlo (MCMC) which samples from the joint probability distribution and use those samples to estimate the hidden variables and parameters. We refer to this algorithm *MCMC-C* (the extra C is for confusion). In order to use MCMC we need to specify all conditional probability distributions (CPDs) for each variable of our model. Below we present the CPDs we use for the inference.

First, we define a CPD for an item $i \in I$. According to our data model, we estimate the probability of item i having true label l_i^* using a multinomial distribution:

$$P(l_i^* \mid \Psi_k, C_k^{-i}, A, G_k^{-i,s}, \pi_k) \propto \prod_{\psi_s^j \in \hat{\Psi}^i} A_s^{\delta(\psi_s^j, l_i^*)} (1 - A_s)^{1 - \delta(\psi_s^j, l_i^*)} \tag{2}$$

where

- l_i^* is true label for item i,

- $\hat{\Psi}^i$ is a set of observations on i including those which were confused with item i (based on the current assignments G),
- ψ_s^j is the observation from worker s on item j,
- $\delta(\psi_s^j, l_i^*) = \begin{cases} 1 & \psi_s^j = l_i^* \\ 0 & \psi_s^j \neq l_i^* \end{cases}$ is the Kronecker delta function that takes the value of either 0 or 1,
- C_k^{-i} is a set of items in cluster C_k except item i,
- $G_k^{-i,s}$ is a set of confused observations G_k except confusion $G_{i,s}$.

Intuitively, we sample a true value for i based on our current knowledge of worker accuracy and observation confusions. Each observation is independent and thus we compute a product of either accuracy of worker A_s if the observation coincides with the item label l_i^* or $(1 - A_s)$ otherwise.

Similarly, we sample confusion variables from a Bernoulli distribution using the following probabilities (if worker s made a confusion during measuring the label of item i):

$$P(G_{i,s} = 0 \mid \psi_s^i, C_k, A_s, \pi_k) \propto (1 - \pi_k) A_s^{\delta(\psi_s^i, l_i^*)} (\frac{1 - A_s}{|L_i|})^{1 - \delta(\psi_s^i, l_i^*)} \quad (3)$$

$$P(G_{i,s} = 1 \mid \psi_s^i, C_k, A_s, \pi_k) \propto \frac{\pi_k}{|C_k| - 1} (\sum_{j \in C_k, i \neq j} A_s^{\delta(\psi_s^i, l_j^*)} (\frac{1 - A_s}{|\bigcup_{j \in C_k} L_i| - 1})^{1 - \delta(\psi_s^i, l_j^*)})$$

The probability that there is no confusion, $P(G_{i,s} = 0)$, depends on the cluster no confusion probability $1 - \pi_k$ as well as the worker accuracy A_s if the item predicted true label l_i^* the same as the vote ψ_s^i or $\frac{1 - A_s}{|L_i|}$ otherwise (assuming each false label is equally likely). In the case of $P(G_{i,s} = 1)$ we consider all possible confusions within the cluster C_k with equal prior probabilities $\frac{1}{|C_k| - 1}$ and we add A_s if the vote ψ_s^i coincides with l_j^* or $\frac{1 - A_s}{|\bigcup_{i \in C_k} L_i| - 1}$ otherwise. In the latter case we assume that if there is a confusion then the space of false labels consists of all unique votes within a cluster minus 1 (true label). We sample the accuracy of workers from a Beta distribution with the following parameters:

$$A_s \sim Beta(\alpha_0 + N_+^s, \alpha_1 + N_-^s) \quad (4)$$

where

$$N_+^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s}) \delta(\psi_s^i, l_i^*) + \frac{G_{i,s}}{|C_k| - 1} \sum_{j \in C_k^{-i}} \delta(\psi_s^i, l_j^*) \quad (5)$$

$$N_-^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s})(1 - \delta(\psi_s^i, l_i^*)) + \frac{G_{i,s}}{|C_k| - 1} \sum_{j \in C_k^{-i}} (1 - \delta(\psi_s^i, l_j^*)) \quad (6)$$

where Ψ^s is a set of votes given by worker s. N_+^s is the number of times a worker s gave the same votes as the corresponding item true label. In the case of a confusion we assume that it measures any of the other $|C_k| - 1$ items at the same probability. N_-^s is the count of worker s observations when it reported a false label (with the similar way of counting confusions).

Algorithm 1: MCMC-C Inference Algorithm

Input: Ψ, α, β, γ
Output: $\{l_i^*\}, \{G_{i,s}\}, \{\pi_k\}$

(1) Initialize A_s, $G_{i,s}$, l_i^* and π_k by drawing from the priors
(2) **for** *iter* **to** *iter_num*
(3) **foreach** $i \in I$
(4) draw $l_i^* \sim multi(\prod_{\psi_s^j \in \hat{\Psi}^i} A_s^{\delta(\psi_s^j, l_i^*)} (1 - A_s)^{1 - \delta(\psi_s^j, l_i^*)})$
(5) **foreach** $G_{i,s} \in G$
(6) draw $G_{i,s} \sim \pi_k A_s^{\delta(\psi_s^i, l_i^*)} (1 - A_s)^{1 - \delta(\psi_s^i, l_i^*)}$
(7) **foreach** $\pi_k \in \pi$
(8) $N_{conf}^k = \sum_{G_{i,s} \in G^k} G_{i,s}$
(9) $\overline{N_{conf}^k} = \sum_{G_{i,s} \in G^k} (1 - G_{i,s})$
(10) draw $\pi_k \sim Beta(\gamma_0 + N_{conf}^k, \gamma_1 + \overline{N_{conf}^k})$
(11) **foreach** $A_s \in A$
(12) $N_+^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s}) \delta(\psi_s^i, l_i^*) +$
(13) $+ \frac{G_{i,s}}{|C_k| - 1} \sum_{j \in C_k^{-i}} \delta(\psi_s^i, l_j^*)$
(14) $N_-^s = \sum_{\psi_s^i \in \Psi^s} (1 - G_{i,s})(1 - \delta(\psi_s^i, l_i^*)) +$
(15) $+ \frac{G_{i,s}}{|C_k| - 1} \sum_{j \in C_k^{-i}} (1 - \delta(\psi_s^i, l_j^*))$
(16) draw $A_s \sim Beta(\alpha_0 + N_+^s, \alpha_1 + N_-^s)$

Finally, we sample the cluster confusions from a Beta distributions using the counts of confused and not confused labels (i.e., using $G_{i,s}$):

$$\pi_k \sim Beta(\gamma_0 + \sum_{G_{i,s} \in G_k} G_{i,s}, \gamma_1 + \sum_{G_{i,s} \in G_k} (1 - G_{i,s})) \quad (7)$$

Given all CPDs presented above the MCMC-C inference algorithm is described in Algorithm 1. As input the algorithm takes a set of observations Ψ and priors, and outputs the probabilities of item labels to be true $\{l_i^*\}$ along with estimates of confusion variables $\{G_{i,s}\}$ and $\{\pi_k\}$. First, we randomly initialize all variables of our model by drawing values from the respective prior distributions (line 1). (In practice, we have found that the MV output is a good initial guess for l_i^*.) Lines 2-6 are sampling from the joint probability distribution of our model where *iter* is the iteration number and *iter_num* is the total number of iterations. We sample labels for $i \in I$ in lines 3-4. For each crowd vote we updated G variables where we also incrementally update count N_{conf}^k and $\overline{N_{conf}^k}$ (lines 7-9). We draw the cluster confusion probabilities by using a posterior Beta with new counts of confused and not confused observations as in Formula 7 (line 10). Then for each worker we initialize its positive and negative counts as they defined in Formula 5 and 6 (lines 11-15). Having N_+^s and N_-^s, we draw the accuracy of each worker from the corresponding posterior Beta distributions (line 16).

Notice that the complexity of Algorithm 1 is proportional to the number of iterations, observations and items in clusters: $O(itern * |\Psi| * max(|C_k|))$ The Algorithm 1 computation overhead (with respect to non confusion aware methods) comes from the necessity to visit all the items within a cluster when we compute counts for A_s and i, i.e., for each item, we take into account the crowd observations which were switched *from* and *to* it. However, in practice the $max(|C_k|))$ doesn't affect the overall computational cost a lot since there are not many items within a cluster and totally not all items are involved into clusters.

5. EXPERIMENTS

We run experiments with the goal of assessing the algorithms in terms of its ability to i) *detect* confusion and ii) *correct* them. We then show a simple but effective way to semi-automatically *prevent* confusions, once detected.

5.1 Datasets

We evaluate the model and algorithm first with a synthetic dataset, to assess performances as we vary the characteristics of the data, and then via nine crowdsourcing experiments of different nature and difficulty. We make publicly available the datasets in the following GitHub repository[1]. We then run four crowdsourcing experiments asking workers to label photos of flags, faces of celebrities, food images, and to assign the title of a movie given the plot. The first three datasets are based on images while the fourth is on text labeling. All tasks where run on the *FigureEight* platform. Overall we collected more than 5100 votes in the tasks. Table 2 shows task and data statistics such as number of items, workers, votes, and proportion of confused votes with respect to entire amount of collected crowd observations.

5.1.1 Synthetic dataset

The synthetic dataset is obtained via a custom synthetic data generator that allows us to vary the number of items, sources (workers), labels (classes), the average accuracy of sources, the probability of confusion for the clusters as well as other parameters. As an input we provide the number of items ($|I|$), the number of sources (n), the number of distinct values per item (V), the average accuracy of sources (A), the probability of confusion (π) and the density (ρ).

We start by generating a dataset without confusions. For each item source pair we (i) run a Bernoulli trial with the probability of ρ to decide whether a source provide an observation on item; if it so then we (ii) run another Bernoulli trial with the probability of A to decide whether a source provided a true value, if not then we choose a false value at random (with the probability $1/(V-1)$). In the next, we inject confusions into the dataset. We form clusters by pairing all items at random (having $|I|/2$ clusters). We revisit all observations and run a Bernoulli trial with the probability of π to decide whether the given observation is confused. If so, then we switch the item of the observation into some other item in a cluster.

5.1.2 Faces

For this experiment, crowd workers were asked to fill out a survey where they need to provide the names of celebrities given their pictures. We built 24 binary clusters of lookalike celebrities and created a survey which have the pictures of celebrities with 6 options where one answer is correct, one answer is from the other item in the cluster (a lookalike celebrity), one answer is *"I don't know"*, and three other answers are just celebrities of the same gender, race and age. Figure 3 is a screenshot of one of the questions in the survey. The confusions were observed on 30 from 48 items, or in 22 from 24 clusters. We found that between items within the same cluster it might be different frequency relations in terms of confusions, i.e., one item might be confused more often than others. In this regards, we revealed 14 one-directed clusters (always the same item was confused) and 8 bi-directional clusters, where confusions were on both items.

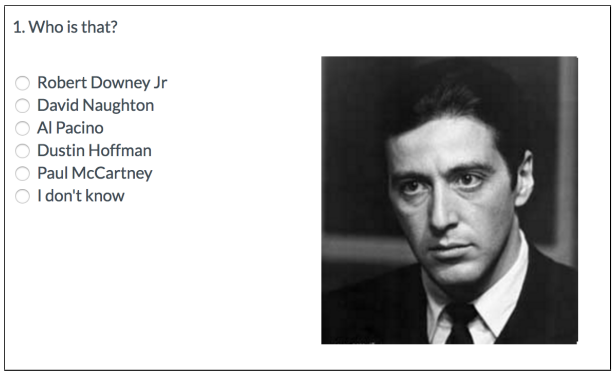

Figure 3: Crowdsourcing task to collect labels on Faces.

5.1.3 Flags

In this job, workers were presented with photos of flags of 60 countries and multiple choice answers. For each picture of a flag, the task is to select the country that has that flag from the multiple-choice answers, in case if a worker is not sure about a country name, he could look at the list of all countries and their' flags provided by us. For each picture we collected 16 votes per image on average. 220 distinct workers took part in the flags task, with an overall accuracy of 90%. We used in a total of 8 test questions to screen out not qualified workers and paid 1 cent per annotating an image, overall contributors satisfaction was 4.4 out of 5.

We then selected initial hypotheses of pairs of labels that we thought could be source of confusion, such as *Moldova* and *Romania*. We selected 13 such clusters, and the errors we define to be of confusions (that is, between items that are in the same cluster) are 7.8% and involve 10 out of the 13 clusters (notice that nearly all errors are confusions). Some confusions are one-directed (e.g., Moldovan flags labeled as Romanian) and others bi-directional (Ireland and Ivory coast labels are swapped).

5.1.4 Food

In this experiment, workers were given with 76 photos of food/dishes and multiple choice answers. The task is to select the right food/dish from the list of options provided. We used in a total of 4 test questions to screen out not qualified workers and paid 1 cent per annotating an image, overall contributors satisfaction was 5 out of 5. As above, we then selected initial hypotheses of pairs of food that we thought could be source of confusion, such as *Artichokes*

[1]https://github.com/Evgeneus/fuzzy-data-fusion

Table 2: The statistics for crowdsourced datasets.

Task Property	Flags	Faces	Food	Plots	Review	Adult	Duck	Sentiment	Dogs
num. of classes	81	72	45	111	3	4	2	4	4
num. of workers	220	21	177	122	135	269	39	27	109
num. of items	100	48	76	100	1500	333	108	584	807
num. of votes	1600	349	1220	1937	7440	3324	4212	5242	8070
num. votes per item (avg ± std)	16 ± 5	7 ± 3	16 ± 10	19 ± 1	7 ± 0	12 ± 3	41 ± 0	11 ± 17	12 ± 0
num. votes per worker (avg ± std)	7 ± 5	16 ± 6	7 ± 4	16 ± 12	55 ± 171	12 ± 23	108 ± 0	194 ± 226	74 ± 88
% of confused votes	7.8%	19.2%	23.8%	6.1%	Unknown	Unknown	Unknown	Unknown	Unknown
workers' accuracy	90%	70.2%	68.8%	90.2%	57%	65%	64%	60%	70%

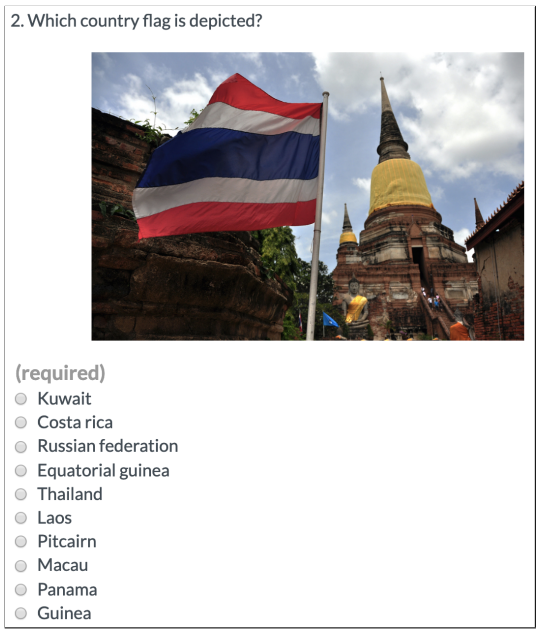

Figure 4: Crowdsourcing task to collect labels on Flags.

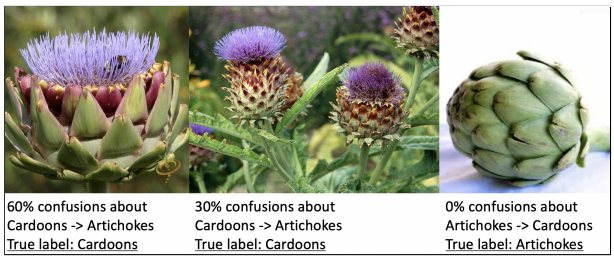

Figure 5: Example of one-directional cluster in Food data

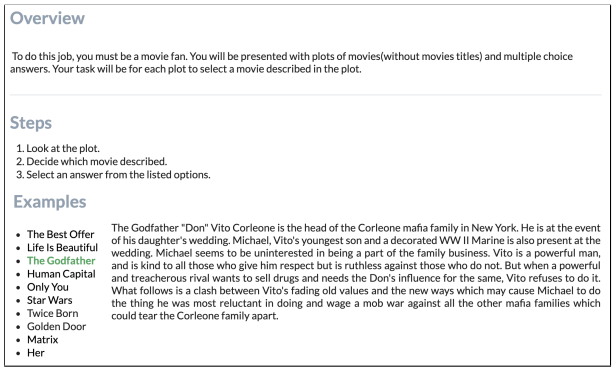

Figure 6: Crowdsourcing task to collect labels on Plots.

and *Cardoons*. In this experiment, we observed one- and bi-directional clusters, such in Figure 5 we demonstrate that people often confuse *Cardoons* with *Artichokes* (60% and 30% of crowd votes in our experiment), but not vice versa.

5.1.5 *Plots*

To create this dataset, we build 50 binary clusters of movies with similar plots. Then we manually collected plots of the selected 100 movies, thus workers were needed to annotate textual data. During the experiment, crowd workers were presented with plots of movies (without movie titles) and multiple choice answers of real movie titles. Workers asked to select a correct movie described in the plot. The instructions for this task is depicted in Figure 6. We used in a total of 10 test questions to screen out not qualified workers and paid 1 cent per annotating a movie plot, overall contributors satisfaction was 4.2 out of 5.

5.1.6 *Other public crowdsourcing datasets*

We further employed five more open-sourced crowdsourcing datasets (from papers published in major conferences) for which we have no specific indication that they suffer from possible confusions and where we have no prior knowledge on confused items, if any. The main motivation for having these datasets is to study if MCMC-C gets worse accuracy in absence of confusions or of well-formulated hypothesis of confusion (clusters). Therefore, we have the following additional datasets (see Table 2 for statistics on these datasets):

Adult. Each task contains a website URL and workers requested to provide the adult content level of the website according to four categories: General Audience, Parental Guidance, Restricted, and Porn [2].

Sentiment. The task is to identify the sentiment for a given face image: neutral, happy, or angry [1]

Review. The task is to provide a sentiment label for a movie review: positive, negative, or neutral review [37].

Dog. The task is to identify a breed of a dog given a picture in one out of four categories [48].

Duck. Workers were asked to recognize whether the image includes a duck or not [42].

5.2 Confusion Detection

We start by testing the ability of the algorithm to detect confusions, our primary goal. Specifically, we are interested in the average probability of confusion detection, that we call *Accuracy* of confusion detection. It is computed as follows:

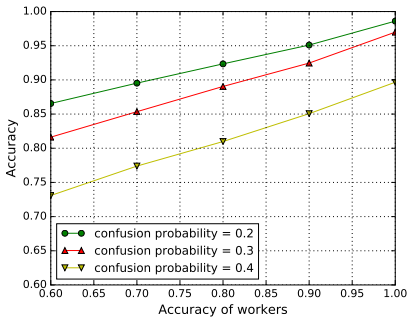

Figure 7: Accuracy of confusion detection for varying workers accuracy.

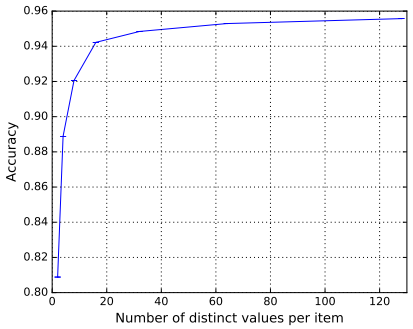

Figure 8: Accuracy of confusion detection as the number of classes grows.

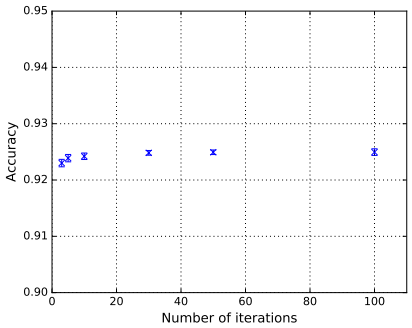

Figure 9: Convergence of MCMC-C with the growing number of iterations.

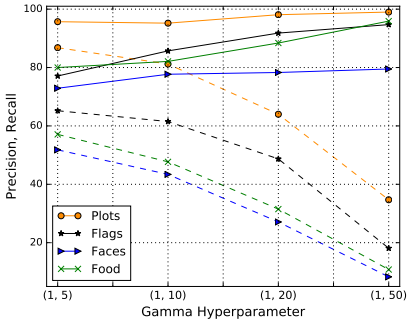

Figure 10: Precision (straight) and Recall (dashed) of confusion detection as Gamma prior of MCMC-C change.

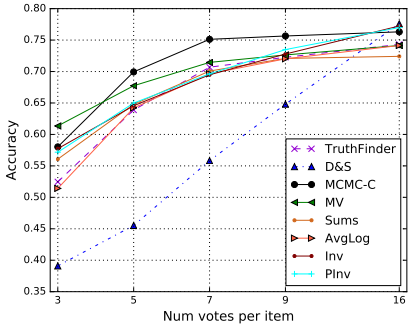

Figure 11: Aggregation Accuracy as the number of votes per items change - Food dataset.

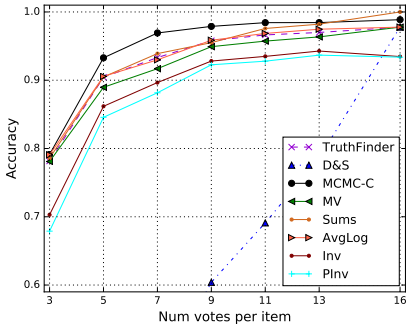

Figure 12: Aggregation Accuracy as the number of votes per items change - Plots dataset.

Table 3: Confusion detection results on real-world datasets.

All data

Dataset	Accuracy	Precision	Recall
Plots	99+-0%	100+-0%	83+-6%
Flags	97+-1%	97+-8%	67+-8%
Faces	87+-1%	83+-6%	39+-6%
Food	86+-1%	98+-3%	40+-5%

5 votes per item

Dataset	Accuracy	Precision	Recall
Plots	98+-1%	98+-6%	64+-11%
Flags	95+-2%	92+-9%	49+-13%
Faces	83+-2%	78+-8%	27+-7%
Food	85+-2%	88+-9%	32+-7%

$$Accuracy = \frac{\sum_{G_{i,s} \in G} [G_{i,s} = G^*_{i,s}]}{\mid G \mid}, \quad (8)$$

where $[G_{i,s} = G^*_{i,s}]$ outputs 1 if the condition holds else 0, $G^*_{i,s}$ is a correct value for a confusion variable $G_{i,s}$, i. e., whether or not a confusion appeared, and $\mid G \mid$ normalizes the metric by the total number of votes given on items that belong to the clusters, so that it is in the $[0, 1]$ interval.

Figure 7 shows the results of applying MCMC-C while varying the accuracy of workers A and probability of confusion π. Our default parameters are 30 workers with average accuracy 0.9, 5000 items, and 50 classes. As expected the output accuracy grows linearly with the worker accuracy. Moreover, the higher A and reasonably low π the MCMC-C gives more accurate results, particularly, for $A = 1$ and $\pi = 0.2$ the accuracy of identifying confusions is 98%. Figure 8 shows instead that accuracy grows with the number of possible classes (here shown keeping $\pi = 0.2$), as it becomes easier to separate confusions.

On crowdsourcing experiments (Table 3), confusion detection accuracy is also fairly high especially in terms of precision, what means we have just a few false positive decisions about confusion identification. Numbers are lower for Food dataset as the overall workers accuracy is lower for that case. Along with the results based on full datasets ("All data"), which include many votes per item (see Table 2) and these numbers are much higher than typical crowdsourcing tasks, Table 3 also shows results considering only 5 randomly selected votes per item - and we discuss later how performances vary with the number of votes per item.

The numbers reported in Table 3 are obtained by averaging the results of 30 repetitions of the experiments.

5.3 The Effect of Hyperparameters

In this section, we study the effect of hyperparameter (α, β, γ) choice for the resulting accuracy in confusion detection. The γ hyperparameters represent our prior knowledge about confusions in a cluster C_k. Assuming that the proportion of confusions is small relative to all votes in the cluster, we tested the following γ priors (the pairs denote α and β parameters in a Beta distribution used to model the a priori belief on confusion): (1, 5), (1, 10), (1, 20), (1, 50), corresponding to prior beliefs over the mean values of confusions, respectively, 0.17, 0.09, 0.048, and 0.019.

Figure 10 displays the resulting MCMC-C Precision and Recall of confusion detection with respect to the different γ priors. As the figure shows, the higher the prior γ_0 value, the higher is Precision and the lower is Recall of confusion identification. We are particularly interested in high Precision since part of what the algorithm does is to correct confused votes so we do not want to do erroneous corrections (i.e., we do not want to harm vote aggregation results). For all of the priors above, the vote aggregation results after confusion correction are better than the original data fusion algorithms and with a 0-3% range in accuracy (depending on the algorithm). For all the following experiments we report results with γ prior of (1, 20) as this value yielded better vote aggregation accuracy.

The other hyperparameters α and β, corresponding to prior beliefs about the accuracy of workers and about the distribution of labels, are common to many classification algorithms and are not specific to confusion detection. In absence of prior knowledge, they can be estimated using test runs (see e.g., [20]) and we do not discuss them further. In the following we set a fixed α prior values of (4, 1) ($\mu = 0.8, std = 0.16$) to all our experiments and other truth discovery models (as it is reasonable to set α_0 higher than α_1 [46]), and adopt a uniform prior for the β hyperparameter, assuming no prior knowledge on the label distribution.

5.4 Confusion Correction

Our secondary goal is to leverage MCMC-C to improve classification accuracy once votes have been collected. The proposed MCMC-C attempts at identifying which votes in a cluster are confused, therefore enabling error correction (reassigning confused votes to correct items), at least in the case of clusters of two labels discussed here. The impact of error correction on the overall classification accuracy also depends on the classification algorithm adopted. We therefore experiment with a set of common vote aggregation algorithms and test the effect on adding confusion correction on top. Specifically, first we correct confusions and then run vote aggregation algorithms. We compare the following algorithms: Majority Voting (`MV`), `TruthFinder` [45] as well as with a base MCMC version such as MCMC Sampling Algorithm (`MCMC`), Dawid-Skene (`D&S`) [8], Sums Hubs and Authorities (`SUMS`) [19], Average Log (`AvgLog`) [16], Investment, and PooledInvestment (`Inv, PInv`) [16], Learning from Crowd (`LFC`) [35], `CRH` [5, 25], `ZenCrowd` [9], `CATD` [24], `GLAD` [43]. Finally, the proposed MCMC with Confused Observations (`MCMC-C`) is an implementation of Algorithm 1 from Section 4.

To evaluate algorithms, we measure vote aggregation *Accuracy* as assigned labels by an algorithm to the true class, and average across all items with the presence of conflicting votes (thus, we do not overestimate the performance of algorithms). To assess the improvements after confusion correction, we compare results before and after correcting for what MCMC-C believes to be confusion errors.

Table 4 shows the improvement in vote aggregation Accuracy for the real-world crowdsourced datasets. The results demonstrate that applying the correction has very large effects in terms of improving accuracy of aforementioned truth discovery algorithms.

Figures 11 and 12 show how performances vary with the number of votes per items for the datasets where D&S is one of the *best* in the full dataset. The reason why D&S does not give good results with low numbers of votes per item is that in these datasets we have rather large number of classes (from 45 in *Food* to 111 in *Plots*), and in this case confusion matrices have a large number of cells (over 10K for *Plots*), so that constructing such matrices (per worker) accurately requires more data. In this case, as the number of data points increases, D&S performs well in the *plots* dataset despite the large number of classes also because the workers' accuracy is high and the confusion is low, the lowest in our datasets. In general, we observe that with number of votes that are in line with typical scenarios, correcting for detected confusions before aggregating results in very much improved accuracy with respect to performing aggregation without correction.

Table 4 and Table 5 depict the results on all datasetas and algorithms. The numbers reported are obtained by averaging the results of 30 repetitions of the experiments.

5.5 Convergence and Scalability of MCMC-C

In this section, we present our study of the convergence and scalability of MCMC-C on the synthetic data. Figure 9 shows the accuracy of confusion detection for the following numbers of iterations: 3, 5, 10, 30, 50, 100. We found that to achieve a sufficiently high accuracy we need only 10 iterations and further increasing the number of iteration doesn't lead to a noticeable improvement in the accuracy.

We analyze the scalability of MCMC-C, where we aim to study how the modeling of confusions affects the time need for inference. To this end, we scale the number of confusions by increasing the number of items which have confused observations (i.e., belong to the clusters). We observe that as we scale the number of items the computational time of MCMC-C stays almost the same with an increase at 10000 items (e.g., when TruthFinder takes 150 sec, MCMC-C consumes 200 sec for the dataset of 1000 items and 500 clusters). Note that the confusion-based algorithms take time equal to its baselines plus the time needs to resolve confusions. In practice, we envision that the number of items with confusions will not be large and therefore we conclude that MCMC-C easily scales up to millions of items.

5.6 Confusion Prevention

A main reason to detect confusion early in the process is the possibility of preventing it. MCMC-C can detect confusions even with a small number of votes per item. Once confusion is detected between a pair of labels, a simple and automated prevention mechanism is to modify the crowdsourcing task by pointing out the possibility of confusion every time the worker selects a label in this pair.

We show an example of this in Figure 13, where once the flag of Haiti or Liechtenstein is selected, a popup automatically appears showing the similarity (and therefore the possible confusion) and asking for verification. We run the same crowdsourcing experiments over flags, plots and food experiments with the prevention mechanism and reduction in confused labels are from 6.1% to 2% for plots, 23.8% to 19.3% for food, 7.8% to 2.6% for flags.

6. CLUSTERING AND COMPARISON TO CONFUSION MATRIX ESTIMATION

Figure 13: Confusion prevention crowd task.

6.1 Iterative and Greedy Cluster Recognition

One of the challenges for MCMC-C is that it takes as input predefined clusters of items. Previously we assumed that clusters are defined by the task designer either manually or via rules that capture a potential confusion. In case a task designer does not have prior opinion about clusters we need to find a way to build them automatically. We also need to analyze the impact of "bad" clusters on MCMC-C confusion detection as well as vote aggregation accuracy. Since MCMC-C is efficient, a trivial way to build clusters is to iterate over possible pairs of labels, estimating confusions at each iteration with MCMC-C, keep clusters of labels identified as confused and re-cluster the others differently. We tested this for our crowdsourced datasets and identify confusions with a percentage very close to what reported for manually selected ones. In the case of randomly created clusters, we obtained accuracy from 96% to 99% on the different datasets (average of 30 repetitions). Note that such high accuracy in confusion detection can be due to the fact that randomly created clusters might have a few confusions and MCMC-C easily handles it and especially in identifying not confused votes. While this works fine in the majority of real-life problems where the number of classes is in the single or double digit range and we look for clusters of size 2, it becomes computationally expensive in other cases.

Therefore, we propose a simple MV-based greedy algorithm for building clusters of items (Algorithm 2). It takes crowdsourced votes and items as input, and outputs clusters of items that are estimated to be at risk for confusion. First, we run the Majority Voting algorithm over the input data (line 2). Then, with the use of MV on results, for each item i we search for the most likely label to be true and the second likely label or $None$. The next step is to retrieve candidate items to form a cluster with the current item i. To this end, we select all items that have the true label (assigned from MV) equal to the second likely label for the current item i (line 6). Finally, we randomly choose an item from $candidates$ items and form a cluster with i (lines 7-8).

Table 5 shows the improvement in vote aggregation Accuracy for the crowdsourced datasets with clusters built via Algorithm 2. We observe that for DS we always improve aggregation accuracy by a large margin, MCMC-C itself provides competitive or better results than original algorithms.

Algorithm 2: Greedy search for clusters

Input: Crowd votes Ψ, I
Output: Set of clusters C

(1) $C = \{\}$
(2) run MV over Ψ
(3) **foreach** $i \in I$
(4) $\quad l_i^* =$ first most common label on i from MV
(5) $\quad l_i^{conf} =$ second most common label on i or $None$
(6) $\quad candidates =[j$ for All $j \in I$ where $l_j^* = l_i^{conf}]$
(7) $\quad i_{conf} =$randomly sample item from $candidates$
(8) $\quad C.append([i, i_{conf}])$
(9) **return** C

6.2 Cluster identification with Machine Learning algorithms

In this section, we aim at testing the effect of clusters created via machine learning algorithms. To this end, we adopt the unsupervised K-Means clustering algorithm on our real-world datasets. For the Plots dataset, which is text-based, we applied K-Means clustering on TF-IDF features extracted from the textual description of the plots. In this case, K-Means takes data (transformed items with TF-IDF vectorizer) and the desired number of clusters, which we set to 50. As a result, we received the clusters identical to the manual ones.

Next, we investigate the effect of clustering image-based datasets (Flags, Food, Faces) with K-Means algorithms. As features for images, we tried i) colored image pixels, ii) colored image pixels + identified edges in the image (using the Prewitt operator[2]). As a result, for Food and Faces datasets we only found some clusters that correspond to the manually identified ones, while for Flags *all* clusters were wrong. These results came from the fact that the algorithm clustered images according to irrelevant information, e.g., in Flags some clusters created based on the presence of sky in photos. Therefore, for images, we need more sophisticated algorithms or/and feature extractors.

Notice that what happens in the case of "bad" clusters is that the algorithm does not detect confusions, but it causes no "harm" in terms of crowdsourced classification with respect to other algorithms such as D&S. Again, code, datasets, detailed experiments results and additional charts are available from the cited Github repo. Our future work will include the application of pre-trained deep networks aimed at assessing similarity.

[2]https://en.wikipedia.org/wiki/Prewitt_operator

Table 4: Accuracy (in %) of Original (*Orig*) and after confusion correction data fusion algorithms on real-world datasets. (*Man* - correction with manually identified clusters, *Grd* - with clusters identified by Algorithm 2.)

Algorithm	Faces			Flags			Plots			Food			Sentiment		Review		Duck		Adult		Dog	
	Orig	Man	Grd	Orig	Man	Grd	Orig	Man	Grd	Orig	Man	Grd	Orig	Grd	Orig	Grd	Orig	Grd	Orig	Grd	Orig	Grd
MCMC-C	-	81	79	-	90	89	-	99	98	-	76	74	-	55	-	72	-	77	-	68	-	78
MV	71	79	74	88	90	89	98	99	98	73	76	74	55	56	69	72	77	77	67	69	78	80
TruthFinder	79	82	80	89	90	89	98	99	98	74	76	74	53	55	71	72	58	77	68	68	81	78
D&S	43	53	47	74	81	89	98	98	98	78	82	75	55	55	74	73	90	77	71	68	83	79
GLAD	73	80	73	89	90	89	98	99	98	74	76	74	55	55	72	74	74	77	67	68	79	79
CRH	78	82	77	88	90	89	98	99	98	73	76	74	49	55	63	70	77	74	69	69	79	81
LFC	43	49	44	61	60	62	78	72	80	55	60	58	58	56	72	75	90	76	73	70	83	81
CATD	79	82	80	88	90	89	98	99	98	74	76	74	53	55	68	74	80	75	68	69	79	81
ZenCrowd	78	81	74	89	90	89	98	99	98	74	76	74	54	56	72	75	58	75	68	69	81	81
MCMC	80	82	79	89	90	89	98	99	98	74	76	74	53	55	71	72	72	77	68	68	80	78
SUMS	78	82	77	91	91	90	100	100	100	72	76	74	54	55	74	72	77	77	65	68	76	77
AvgLog	78	81	77	89	90	89	98	99	98	74	76	74	55	55	73	72	77	77	69	68	78	78
Inv	81	82	83	87	89	89	93	99	98	77	78	75	45	54	67	69	77	78	71	67	80	78
PInv	81	84	82	87	90	89	93	99	98	78	78	75	46	54	67	69	78	78	72	68	80	78

Table 5: Accuracy (in %) of Original and after confusion correction data fusion algorithms on real-world datasets with only 5 votes per item.

Data 5 votes item

Algorithm	Faces			Flags			Plots			Food		
	Orig	Man	Grd	Orig	Man	Grd	Orig	Man	Grd	Orig	Man	Grd
MCMC-C	-	77	77	-	80	78	-	94	90	-	71	66
MV	68	73	70	77	81	78	87	93	87	67	72	68
TruthFinder	77	78	78	77	80	78	89	94	89	65	69	66
D&S	41	49	44	51	63	54	33	59	41	45	57	48
GLAD	72	76	73	77	80	78	88	92	86	67	71	68
CRH	74	77	76	77	81	78	87	92	87	67	71	68
LFC	36	37	38	29	27	31	13	13	14	22	22	23
CATD	76	78	77	77	81	78	87	93	86	65	71	68
ZenCrowd	74	76	74	77	81	78	88	93	86	66	71	67
MCMC	75	77	76	78	80	78	89	93	90	66	70	66
SUMS	73	77	73	77	80	78	89	93	90	64	69	66
AvgLog	73	78	73	77	80	78	89	94	90	64	68	65
Inv	73	78	76	75	79	76	83	90	87	64	79	65
PInv	72	78	76	74	80	76	82	90	86	66	70	65

6.3 Comparison with D&S Confusion Matrix

We have shown that MCMC-C is an effective to detect confusion over a variety of datasets, and that once confusion is detected, it can then be prevented to a large extent with simple modifications to the task. The algorithm scales well to large datasets (both in items and workers).

Part of our work included a comparison with D&S as a baseline, and of what happens when we preprocess confusions and then apply D&S. We are specifically interested in D&S as a main representative of algorithms that do aim at identifying confusion matrices and therefore in a way do deal with confusions. However, D&S per se does not output classes that some workers may confuse, but it "simply" produces estimated labels and per-worker confusion matrices.

In principle it is possible to use such matrices to determine pairs (or cliques) of classes that can be source of frequent confusions, and this requires a way to map the set of worker-specific matrices into sets of possibly confused labels. However, how to do this is not obvious and requires further research. Simply taking matrix cells off the diagonal where the average value is high (denoting that workers frequently mistakes the estimated correct label - in the row - with the voted label - in the column) often leads to poor results especially in the conditions discussed above (large number of cells and relatively sparse matrices with workers votes). If workers' accuracy is also not very high, then D&S has a hard time distinguishing errors and confusions.

To show how D&S behaves in the presence of classes that some workers confuse, we plot in Figure 14 three *confusion matrices histograms* obtained by D&S, corresponding to three (simulated) experimental conditions. Each matrix cell contains an histogram plotting the percentage of workers (y axes) that have a specific confusion value for that cell (the X axis is divided in buckets of size 0.1). So for example, cell [1,2] (first row, second column) shows a histogram denoting how many workers have a given confusion value for that cell. The expectation is that cells on diagonal have bars close to 1, and off-diagonal cells have most of the mass towards zero. All matrices correspond to experiments with 5 classes, workers' accuracy in the [0.7-0.9] range, and have a varying percentage of workers that confuse class 1 and 2. Therefore, we assume to observe signals for confusions in cells [1,2] and [2,1]. The dataset for the top matrix has 50% of workers confused, and has 5 votes per item, 25 votes per worker. The one in the middle also has 50% of workers confused, and 3 votes per item, while the bottom one has 5 votes per item but only 10% of workers are confused between classes 1 and 2.

What we can see is that while for the top matrix we could infer confusions, for example by observing that the histogram has mass close to both extreme and not just one (see in particular cell [1,2]), when the number of votes per items goes down (matrix in the center) or the percentage of confused workers is a probably more realistic 10%, making confusion determination becomes challenging. Things get worse as the number of classes increase, as observed earlier.

7. CONCLUSIONS AND LIMITATIONS

The learning we take home from the experiments is that the approach proposed is effective in detecting confusions

across several datasets with different characteristics and is robust to clustering methods and hyperparameters selection.

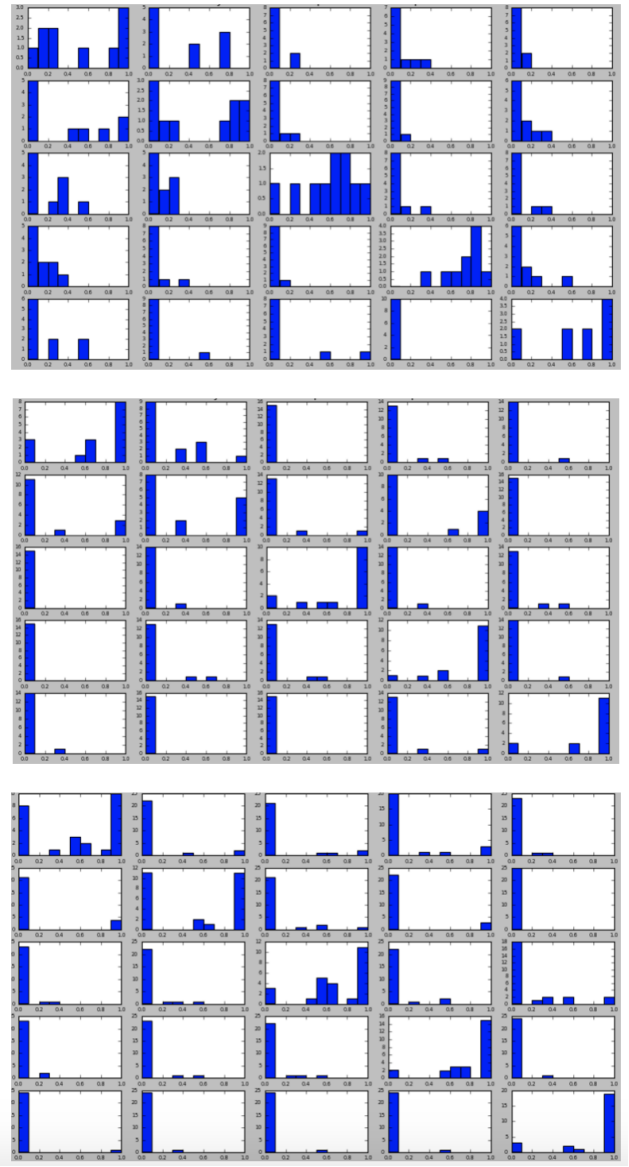

Figure 14: Dawid-Skene for Confusion Detection.

The approach helps in nearly all cases in improving the quality of crowd classification and it does not negatively affects accuracy for problems where there is no confusion. We show that it improves over the "standard" D&S approach, even with automatically generated confusion clusters.

A limitation of the approach so far is still in the number of datasets which, as much as we endeavored to select a "diverse" sample, is still relatively small. We also foresee that improvements are possible in the selection of the initial clusters as well as hyperparameters, possibly leveraging iterative or reinforcement learning based approaches that tune the parameters as crowdsourcing progresses. Indeed, the main limitation of the current work is for the case where the problem is characterized by large number of classes and of confusions among clusters of more than two classes (so that brute force and iterative methods are not applicable), and where the MV-based greedy approach is not effective and task designers do not have intuitions or rules about possible confusions. This is not a frequent problem in practice in crowdsourcing, but it is one for which further work is needed in the cluster identification phase.

8. REFERENCES

[1] http://dbgroup.cs.tsinghua.edu.cn/ligl/crowddata/.

[2] https://github.com/ipeirotis/get-another-label/tree/master/data.

[3] O. Alonso and G. Marchionini. *The Practice of Crowdsourcing*. Morgan Claypool, 2019.

[4] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 3:1–3:13, New York, NY, USA, 2019. ACM.

[5] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas. Crowdsourcing for multiple-choice question answering. In *AAAI*, pages 2946–2953, 2014.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117, 1998.

[7] J. C. Chang, S. Amershi, and E. Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 2334–2346, New York, NY, USA, 2017. ACM.

[8] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C Applied Statistics*, 28(1), 1979.

[9] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, pages 469–478, 2012.

[10] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: the role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[11] X. L. Dong, L. Berti-Equille, and D. Srivastava. Data fusion : Resolving conflicts from multiple sources. In *Procs of WAIM2013*. Springer, 2013.

[12] A. Dumitrache, O. Inel, B. Timmermans, C. Martinez-Ortiz, R.-J. Sips, L. Aroyo, and C. A. Welty. Empirical methodology for crowdsourcing ground truth. *CoRR*, abs/1809.08888, 2018.

[13] C. Eickhoff. Cognitive biases in crowdsourcing. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 162–170, New York, NY, USA, 2018. ACM.

[14] M. Giancola, R. Paffenroth, and J. Whitehill. Permutation-invariant consensus over crowdsourced labels. In *HComp*, 2018.

[15] H. O. Hartley. Maximum likelihood estimation from incomplete data. *Biometrics*, pages 174–194, 1958.

[16] D. R. Jeff Pasternack. Knowing what to believe (when you already know something). *International*

Conference on Computational Linguistics, pages 877–885, 2010.

[17] R. Jin, Y. Dou, Y. Wang, and X. Niu. Confusion graph: Detecting confusion communities in large scale image classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1980–1986, 2017.

[18] V. S. S. Jing Zhang, Xindong Wu. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

[19] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, Sept. 1999.

[20] E. Krivosheev, B. Benatallah, and F. Casati. Crowd-based multi-predicate screening of papers in literature reviews. In *Proceedings of WWW2018*. International World Wide Web Conferences Steering Committee, 2018.

[21] E. Krivosheev, V. Caforio, B. Benatallah, and F. Casati. Crowdsourcing paper screening in systematic literature reviews. In *Procs of Hcomp2017*. AAAI, 2017.

[22] D. Lan, K. Reed, A. Shin, and B. Trushkowsky. Dynamic filter: Adaptive query processing with the crowd. In *Procs of Hcomp2017*. AAAI, 2017.

[23] J. Li, Y. Baba, and H. Kashima. Incorporating worker similarity for label aggregation in crowdsourcing. In V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 596–606, Cham, 2018. Springer International Publishing.

[24] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han. A confidence-aware approach for truth discovery on long-tail data. *PVLDB*, 8(4):425–436, 2014.

[25] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014.

[26] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(2):97–108, 2013.

[27] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16, 2016.

[28] C. Liu and Y. M. Wang. Truelabel + confusions: A spectrum of probabilistic models in analyzing multiple ratings. In *Procs of ICML2012*. ICML, 2012.

[29] V. K. C. Manam and A. J. Quinn. Wingit: Efficient refinement of unclear task instructions. In *Procs of HComp2018*. AAAI Publications, 2018.

[30] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth Discovery on Crowd Sensing of Correlated Entities Categories. *ACM SenSys*, pages 169–182, 2015.

[31] M. L. Mortensen, G. P. Adam, T. A. Trikalinos, T. Kraska, and B. C. Wallace. An exploration of crowdsourcing citation screening for systematic reviews. *Research Synthesis Methods*, 2016. RSM-02-2016-0006.R4.

[32] J. Pasternack and D. Roth. Making Better Informed Trust Decisions with Generalized Fact-Finding. In *IJCAI*, 2011.

[33] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava. Fusing data with correlations. In *SIGMOD*, pages 433–444. ACM Press, 2014.

[34] J. Ramírez, M. Baez, F. Casati, and B. Benatallah. Crowdsourced dataset to study the generation and impact of text highlighting in classification tasks. *BMC Research Notes*, 12(1):1–4, 2019.

[35] 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(4), 2010.

[36] A. T. T. J. N. Robin Ouyang, Mani Srivastava. Truth discovery in crowdsourced detection of spatial events. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):1047–1060, 2016.

[37] F. Rodrigues and F. C. Pereira. Deep learning from crowds. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015.

[39] Y. S. H. K. Satoshi Oyama, Yukino Baba. Accurate integration of crowdsourced labels using workers self-reported confidence scores. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[40] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, June 2015.

[41] X. Wang, Q. Z. Sheng, X. S. Fang, and L. Yao. An Integrated Bayesian Approach for Effective Multi-Truth Discovery. In *CIKM*, 2015.

[42] P. Welinder, S. Branson, P. Perona, and S. J. Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[43] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. pages 2035–2043, 2009.

[44] J. Yang, T. Drake, A. Damianou, and Y. Maarek. Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 23–32. International World Wide Web Conferences Steering Committee, 2018.

[45] X. Yin, J. Han, and S. Y. Philip. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808, 2008.

[46] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han.

A bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB*, 5(6):550–561, 2012.

[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] D. Zhou, J. Platt, S. Basu, and Y. Mao. Learning from the wisdom of crowds by minimax entropy. In *Procs of Nips 2012*, 2012.