HHAI 2023: Augmenting Human Intellect
P. Lukowicz et al. (Eds.)

doi:10.3233/FAIA230072

Designing Closed-Loop Models for Task Allocation

Vijay KESWANI [a,1], Elisa CELIS [a], Krishnaram KENTHAPADI [b] and Matthew LEASE [c]
[a] *Yale University*
[b] *Fiddler AI*
[c] *University of Texas at Austin*

Abstract. Automatically assigning tasks to people is challenging because human performance can vary across tasks for many reasons. This challenge is further compounded in real-life settings in which no oracle exists to assess the quality of human decisions and task assignments made. Instead, we find ourselves in a “closed” decision-making loop in which the same fallible human decisions we rely on in practice must also be used to guide task allocation. How can imperfect and potentially biased human decisions train an accurate allocation model? Our key insight is to exploit weak prior information on human-task similarity to bootstrap model training. We show that the use of such a weak prior can improve task allocation accuracy, even when human decision-makers are fallible and biased. We present both theoretical analysis and empirical evaluation over synthetic data and a social media toxicity detection task. Results demonstrate the efficacy of our approach.

Keywords. task allocation, closed training, crowdsourcing

1. Introduction

Human decision-making is ubiquitous: in the daily life of organizations or “pure” *human computation* settings without automation, in making labeling decisions to train and test AI systems, and in *human-in-the-loop* architectures that dovetail automated AI with human abilities. People are also naturally fallible: some people perform better than others across different tasks due to a wide range of factors (e.g., background or experience), as observed in recruitment [4] and healthcare [39]. Human error can be due to noise (e.g., fatigue/oversight) and systematic patterns of error (e.g., varying skill). Group decisions can also be fallible and systematically biased depending on the composition and decision process. Whereas “wisdom of crowds” [47] can boost collective intelligence via group diversity, lack of such diversity can amplify biases rather than mitigate them [18].

Task allocation (cf. [22]) seeks to optimize the overall quality of outcomes by effectively matching people to tasks. Accurate task allocation has applications in crowdsourcing [13], human-in-the-loop frameworks [28], and collaborative web platforms [1]. A key assumption underlying most prior work on task allocation is that an oracle exists to provide feedback on the quality of human decisions and task assignments made. In real life, however, the same fallible human decisions we rely on must often also provide

[1]Corresponding Author: Vijay Keswani, vijay.keswani@yale.edu.

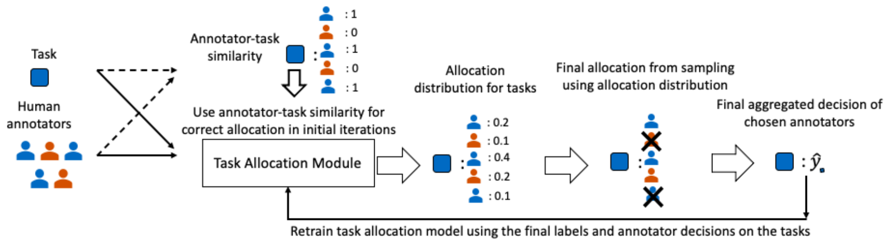
Task
Human annotators
Annotator-task similarity
:1
:0
:1
:0
:1
Use annotator-task similarity for correct allocation in initial iterations
Task Allocation Module
Allocation distribution for tasks
:0.2
:0.1
:0.4
:0.2
:0.1
Final allocation from sampling using allocation distribution
Final aggregated decision of chosen annotators
: $\hat{y}$
Retrain task allocation model using the final labels and annotator decisions on the tasks

Figure 1. A closed-loop task allocation model in which predictions are fed back to train the model. To bootstrap training, we use prior information on human-task similarity, evaluated here by matching task & annotator colors.

the basis for evaluating allocation decisions. When a hiring or admissions committee makes a decision whether to hire/admit a given candidate, all we have is the committee's decision; no outside oracle exists to provide a definitive evaluation of the committee's decision. Similarly, social media content moderation relies on decisions from human moderators. Moreover, decision criteria are often organization-specific model [38]. These applications motivate our investigation of the *closed* training loop setting in which the aggregated annotations from an input-specific selection of human decision-makers are fed back into the system to train the task allocation model (**Figure 1**). However, considering that human decision-makers can make imperfect decisions, the question arises whether their aggregated decisions can be used to train an accurate task allocation framework.

A central challenge with such a framework is how to address human inaccuracy and bias, especially in the initial training iterations. Unsupervised aggregation of human decisions [11] can provide noisy feedback on task allocation efficacy [15, 13, 32, 54, 49]; however such noise, especially in initial training iterations, can result in a slow or non-converging training process. Furthermore, any bias in human decisions may be fed back into task allocation training, further amplifying system error [35] (see §2.1). Particularly problematic are human biases stemming from a lack of background, training, or prejudice, which can consistently impair performance. Another factor that can influence human decisions is underlying demographic identity. Goyal et al. [19] observe that the demographic identity of crowd annotators impacts their toxicity ratings. Consequently, they call for "*ML engineers and researchers to ... consider all the different kinds of specialized rater pools we should be imagining, considering, designing, and testing with.*" Multiple other studies [29, 5, 17, 46, 41] have reported significant differences in ratings across annotator demographics (see §6 for additional related work). Motivated by these studies, we tackle the problem of developing allocation methods that are input-specific, contextually-aware, and cognizant of the background of the human annotators.

Contributions. In this work, we formulate the challenge of closed-loop learning from noisy human labels and propose two online learning algorithms for it. To mitigate inaccuracy from fallible human decision-makers, we propose exploiting a specific form of weak prior on human-task similarity to initialize the task allocation model (§2.2). This enables us to obtain relatively accurate class labels for initial inputs and thereby effectively bootstrap the training process. The first algorithm we present, *Strict-Matching*, directly uses the prior information to initialize the allocation model. The second, *Smooth-Matching*, provides a smoother transition from the prior distribution to learning from noisy feedback during training. We demonstrate the efficacy of our methods via both theoretical analysis (§2.3) and empirical improvement on synthetic and real-world datasets (§3 and §4). The latter extends beyond the classic assumption of universal, objective truth to consider recent advocacy for recognizing subjective, community-based gold standards [44, 29, 17, 19].

2. Model and Algorithm

We consider the binary classification task of predicting label y from input $x \in \mathbb{R}^n$. We assume each input x belongs to one or more categories $z \in \mathscr{Z}$, which could correspond to any demographic or task-specific interpretable feature. Given any input x (i.e., a task), the goal of task allocation is to choose appropriate human annotators (interchangeably referred to as individuals or decision-makers) from a given pool, whose aggregated prediction is the final predicted label for x. Assume there is a pool of m available annotators $e_1, \dots, e_m : \mathbb{R}^n \to \{0,1\}$, with $e_i(x)$ denoting the i-th annotator's prediction for x. For input x, the task allocation model D_u infers a probability distribution over annotators: $D_u : \mathbb{R}^n \to \Delta^m$, where u denotes model parameters and Δ^m denotes the m-dimensional simplex[2]. When possible, we omit subscript u and refer to D_u by D. $D(x)_i$ denotes the model probability assigned to individual i, reflecting the model's estimate of that individual's ability to correctly label input x, relative to the other annotators. While not evaluated in our study, our framework also supports each person having additional input-specific costs associated with their predictions (see discussion of this point in §5).

Committee Voting. Given the task allocation model $D(x)$'s inferred probability distribution over the pool of m annotators, the top-k can be selected to form a committee. When $k > 1$, the committee's decision is determined by majority vote (assuming k is odd, no tie-breaking is required). A technical detail is that we sample annotators with replacement according to $D(x)$ so that the the majority vote of the committee implicitly equals (on average) the weighted majority vote of all m annotators, with $D(x)$ probabilities as weights. Alternatively, one could sample the k annotators without replacement and explicitly weight member votes by $D(x)_i$.

Online learning. Assuming a streaming setting, after each input x is labeled by a selected committee, the (potentially noisy) label is fed back into the closed-loop learning process to update the model $D(x)$. This online learning setting supports potential use in various real-world applications [14, 3]. However, such a noisy feedback loop also risks problematic predictions when trained without care; our algorithms are thus designed to address this.

2.1. *Training the allocation framework*

An ideal training process for an allocation framework learns a partition of the feature space and assigns annotators to those partitions where they are expected to be most accurate. Prior training approaches optimize over labeled datasets to learn an allocation model that simulates such a partition [28, 49, 15]. In this section, we first summarize training procedures from prior work (that assume access to oracle training labels or rewards/penalties). We then discuss extensions of these procedures for closed-loop training.

Prior work training allocation models with gold. Assume input x having group attribute z and true binary label y, $D(x)$ is the task allocation model probability distribution over the pool m experts, and $e_i(x)$ binary prediction of expert i. A general training algorithm, with access to ground truth labels, will update the allocation model to reward the correct experts (for whom $e_i(x) = y$) and penalize those who are incorrect:

$$D(x)_i = \begin{cases} D(x)_i + \delta^{(i)}_{reward}(x,y), & \text{if } e_i(x) = y \\ D(x)_i - \delta^{(i)}_{penalty}(x,y), & \text{if } e_i(x) \neq y \end{cases} \tag{1}$$

[2]distribution over m annotators: $\forall d \in \Delta^m$, $0 \le d_i \le 1$ for all $i \in \{1, \dots, m\}$ and $d^\top \mathbf{1} = 1$.

where $\delta^{(i)}_{reward}(\cdot), \delta^{(i)}_{penalty}(\cdot) : \mathscr{X} \times \{0,1\} \to \mathbb{R}_{\geq 0}$ are input and annotator-specific updates and chosen so that the updated weights sum to 1[3]. This appropriately rewards/penalizes the annotators, yielding allocation model updates that simulate these rewards/penalties.

The reward/penalty functions are constructed by framing the problem as an optimization program. In the case of Keswani et al. [28], rewards/penalties are constructed as follows: given (x,y), allocation parameters u, and committee size k, first select a committee C of k annotators using $D_u(x)$ and compute the (probabilistic) prediction $\hat{y}_u(x)$ by taking the mean of the selected annotators, i.e., $\hat{y}_u(x) := \sum_{i \in C} e_i(x)/|C|$. Then minimize the following regularized log-loss function: $\mathscr{L}_D(u) := \mathbb{E}_{x,y}\left[-y\log(\sigma(\hat{y}_u(x))) - (1-y)\log(1-\sigma(\hat{y}_u(x)))\right]$, where σ is the standard sigmoid function. Expected loss can be computed by the mean over a batch of training samples, with optimization performed via gradient descent. The gradient updates for this loss function can be seen to reward the correct annotators and penalize the incorrect annotators [28]. Hence, functionally, each step of this algorithm has a similar structure as **Equation 1**. Other prior training algorithms can also be shown to have similar underlying reward/penalty structure; see Appendix B in Supplementary Material (§7) for examples.

Training using noisy aggregated human labels. In this work, we focus on the more challenging case of having access to fallible human decisions only, with no oracle feedback regarding their accuracy (i.e., no access to y). Lacking gold labels, one way to directly use the above training process is to learn from noisy, aggregate human labels. Given input x and committee C selected using $D(x)$, the predicted label $\hat{y}(x) := \mathbf{1}\left[\sum_{i \in C} e_i(x)/|C| > 0.5\right]$. Then, the training updates can substitute y with $\hat{y}$ in Equation 1:

$$D(x)_i = \begin{cases} D(x)_i + \delta^{(i)}_{reward}(x, \hat{y}(x)), & \text{if } e_i(x) = \hat{y}(x) \\ D(x)_i - \delta^{(i)}_{penalty}(x, \hat{y}(x)), & \text{if } e_i(x) \neq \hat{y}(x) \end{cases} \tag{2}$$

By substituting true class labels with noisy aggregated labels, existing training allocation algorithms [28, 36] can be used without major changes (e.g., substitute y with $\hat{y}$ in above loss $\mathscr{L}_D(u)$). While simple, this approach also has a potential downside: when the majority of the annotators are consistently biased against any group $z \in \mathscr{Z}$, this unsupervised training process is unable to detect such bias.

Bias propagation when training using noisy labels. Assuming a binary group attribute, we show below that: if (i) the starting allocation model chooses annotators randomly, and (ii) the majority of the annotators are biased against or highly inaccurate with respect to a group attribute type (e.g., a disadvantaged group), then the above training process leads to disparate performance with respect to the disadvantaged group. For $\alpha > 0.5$, assume that α fraction of annotators are biased against group $z = 0$ and $(1-\alpha)$ fraction are biased against group $z = 1$. If a person is biased against $z = j$, they will always predict correctly for inputs with $z{=}1{-}j$ but predict correctly for inputs with $z{=}j$ with probability 0.5.

Lacking an informative prior, training will start with $D(x)$ assigning uniform probability $1/m$ to all m annotators. When $k = 1$, a single person decides the label for input x. In this case, the starting accuracy for group $z = 1$ elements will be $\alpha + 0.5(1-\alpha)$, and for group $z = 0$ elements, $(1-\alpha) + 0.5\alpha$. Therefore, the difference in expected accuracy for group $z = 1$ vs. $z = 0$ elements will be $(\alpha - 0.5)$. The larger the value of α, the greater the disparity will be. Hence, with biased starting allocation model and predicted labels used for retraining, the bias will propagate to the learned model.

[3] i.e., $\sum_{i=1}^{m} \delta^{(i)}_{reward}(x,y) \cdot \mathbf{1}(e_i = y) - \sum_{i=1}^{m} \delta^{(i)}_{penalty}(x,y) \cdot \mathbf{1}(e_i \neq y) = 0$.

Claim 2.1. *In the above setting, the disparity between accuracy for group* $z{=}0$ *and accuracy for group* $z{=}1$ *does not decrease even after training using multiple Eqn. 2 steps.*

The proof is provided in the supplementary material. In §3, we simulate a setting wherein most annotators are biased against certain input categories. Results show that prior training algorithms perform poorly, yielding low allocation accuracy.

2.2. Injecting Prior Information

In real life, no oracle exists to provide us feedback on our fallible or biased human decisions. There is no oracle gold training data to guide initial allocation decisions, nor is there gold feedback on human decisions made during closed-loop training. How then can imperfect human decisions train an accurate task allocation model? Our key insight is to exploit weak prior information on human-task similarity to bootstrap model training.

Motivating Examples. *Example 1.* When a company recruits a new employee, the human decision-makers are typically current employees, and more specifically, a "hiring loop" of employees possessing appropriate expertise to assess the candidate's credentials. Assuming a company knows the varying expertise of its own workforce, prior information exists to match decision-makers to new candidates. In addition, organizations today appreciate the importance of forming hiring committees that combine diversity and expertise [42].

Example 2. In content moderation, moderator decisions vary due to many compatibility factors. For example, a lack of familiarity with the dialect of the content's author can lead to biased decisions [40, 10]. Whether the moderator has themself has been a target of hate speech [29], or whether their own demographic identity aligns with that being targeted in the content they are reviewing [19] can also impact their decisions. Thus, once differences in judging behavior among moderators are acknowledged and accepted, it creates a space for matching different groups of moderators to different content types, based on moderator background (which can be collected via an onboarding questionnaire).

Encoding Prior Information. Any allocation model induces a probability distribution over the decision-makers for each input, such that the probability assigned to each decision-maker represents the confidence in their correctness. An initial approximation of this distribution over the human decision-makers can be derived using the contextual information of the application where the allocation model is being employed. For the motivating examples above, such weak prior information already exists to 1) appoint employees to a hiring loop who are capable of evaluating a candidate (by matching areas of expertise); and 2) select moderators to review content appropriate to their background (by matching target and annotator demographics/dialect).

In absence of labeled training data, we can use this prior information to bootstrap the closed-loop training process. The prior information is encoded in our framework using a similarity function $dSim : \{e_1, \ldots, e_m\} \times \mathscr{X} \to [0,1]$, i.e., specifying a continuous similarity score matching each individual person to each content category. As shown above, starting with a random allocation model is challenging when we also lack oracle feedback on the accuracy of human decisions in the closed-loop training process. Especially problematic are settings when the majority of annotators are biased against certain groups, as observed from the stylized example in Claim 2.1. By starting with some prior information about which people (or groups of people) might be best suited to each type of task (or category of content) using *dSim*, we seek to address this flaw of the closed training framework and bootstrap an accurate training process. Indeed Claim 2.2, shows that using an appropriate *dSim* can address the issues observed in the setting of Claim 2.1.

Claim 2.2. *Revisiting Claim 2.1, suppose $dSim(e_j, z){=}1$ if annotator e_j is unbiased for category z and γ otherwise, where γ is any constant $\in [0,1]$. Consider the allocation induced by this dSim function (i.e., for input (x,z), allocation output $D(x)_i \propto dSim(e_i, z)$). Then the difference between the accuracy for group $z{=}0$ and $z{=}1$ lies in $\left[\frac{\gamma}{2}, \frac{\alpha}{1-\alpha} \cdot \frac{\gamma}{2}\right]$.*

The proof is provided in the supplementary material. Claim 2.2 shows that smaller γ values imply *dSim* is better able to differentiate between biased and unbiased annotators. Hence, the better *dSim* is at differentiating biased and unbiased annotators, the smaller the disparity in performance across groups of the starting allocation model. We thus utilize *dSim* to mitigate biases in training using noisy labels (i.e., **Eq.** (2)). Our proposed algorithms operate on this general formulation of $dSim(e,z)$.

Algorithm 1 Training with prior information.
Input: $(x_1,z_1),\ldots(x_T,z_T)$, humans $e_1,\ldots,e_m$, k
Output: Trained task allocation model

```
1: Set initial allocation $D_0$ s.t. for any input $x$ in category $z$, we have $D_{u_0}(x)_i \propto dSim(e_i, z)$
2: for $t \in \{1,2,\ldots,T\}$ do
3:   $D_{t-1}(x_t) \leftarrow$ Allocation distribution for $x_t$
4:   $C \leftarrow$ Choose committee of size $k$ using distribution $D_{t-1}(x_t)$
5:   $\hat{y}_t \leftarrow$ Aggregated decision of annotators in $C$
6:   $D_t \leftarrow$ Update allocation by training on $(x_t, \hat{y}_t)$
7: return $D_T$
```

2.3. *Training a closed-loop framework using dSim*

Algorithm 1 presents our general training process. The first step ensures that initial allocation follows the prior information provided by *dSim*. Subsequent training steps learn from noisy, aggregated decisions to further improve the task allocation model accuracy. Concrete methods to implement this algorithm are discussed next.

Training Method 1: Strict-Matching. One way to implement Algorithm 1 in practice is to encode the *dSim* function within the initial task allocation model. In particular, we set initial allocation model parameters such that, for the starting allocation model D_{u_0} and input (x,z) and annotator e_i, we have that $D_{u_0}(x)_i \propto dSim(e_i, z)$ (Step 1). This can be feasibly accomplished in most applications using unlabeled data. The rest of the training process is the same as §2.1 and Equation (2): for every input, reward the annotators whose prediction matches with aggregated prediction and penalize those who do not (using gradient of loss $\mathscr{L}_D$). Aggregation of selected annotator predictions can be implemented in various ways; see *Committee Voting* in §2. To add further robustness, we use a batch update process; i.e., for a given integer B, train the model after observing B samples. This approach exploits the *dSim* prior to set the initial $D(x)$ distribution, followed by closed-loop training with noisy aggregate feedback to further improve the task allocation model.

Algorithm Strict-Matching
Input: input stream $(x_1,z_1),\ldots(x_T,z_T)$, experts $e_1,\ldots,e_m$, function *dSim*, batch size B, committee size k, parameter T_d, rate η, loss function $\mathscr{L}_D$.
Output: Trained allocation model parameters u.

```
1: Set initial model parameters $u_0$ s.t. for any input $(x,z)$, we have $D_{u_0}(x)_i \propto dSim(e_i, z)$
2: $S \leftarrow \emptyset$
3: for $t \in \{1,2,\ldots,T\}$ do
4:   $D_{u_{t-1}}(x_t) \leftarrow$ Allocation output for $x_t$
5:   $D(x_t) \leftarrow D_{u_{t-1}}(x_t)/\mathrm{sum}(D_{u_{t-1}}(x_t))$
6:   $C \leftarrow$ Sample $k$ annotators from $D(x_t)$
7:   $\hat{y}_t \leftarrow$ Aggregate label of committee $C$
8:   $S \leftarrow S \cup \{(x_t, y_t)\}$
9:   if $|S| = B$ then
10:    $u \leftarrow u - \eta \cdot \frac{\partial \mathscr{L}_D(u)}{\partial u}\Big|_S$
11:    $S \leftarrow \emptyset$
12: return $u_T$
```

Training Method 2: Smooth-Matching. To obtain a better transition from the *dSim* prior to the allocation model learnt during closed-loop training, we can gradually wean ourselves off of the prior by decreasing its relative weight as more observed evidence accumulates.

In other words, the allocation employed at any iteration can be chosen as a convex combination of the allocation encoded by the *dSim* prior and the allocation trained using the observed samples (and aggregated class labels). This method of combining prior and observed data is conceptually similar to Bayesian or Laplacian smoothing techniques [43, 51]. Additive combination yields a task allocation distribution incorporating both the prior distribution and the empirical distribution. The smoothing parameter μ is set to be an increasing function of the number of observations, ensuring that prior information *dSim* is used primarily in the initial training iterations. Full details are provided in Algorithm *Smooth-Matching*. Parameter T_d in Smooth-Matching controls the influence of *dSim* on the training process. The first T_d iterations focus on obtaining accurate labels for initial samples to bootstrap the training process. After T_d iterations, the weight given to the prior is smaller than the weight given to the distribution learned during training.

Algorithm Smooth-Matching

Input: input stream $(x_1,z_1),\dots(x_T,z_T)$, experts $e_1,\dots,e_m$, function *dSim*, batch size B, committee size k, parameter T_d, rate η, loss function $\mathscr{L}_D$.

Output: Trained allocation model parameters u.

```
1:  $S \leftarrow \emptyset$
2:  **for** $t \in \{1,2,\dots,T\}$ **do**
3:    $\mu \leftarrow T_d/(t+T_d)$
4:    $D_{dSim}(x_t) \leftarrow [dSim(e_1,z_t),\dots,dSim(e_m,z_t)]$
5:    $D_{dSim}(x_t) \leftarrow D_{dSim}(x_t)/\text{sum}(D_{dSim}(x_t))$
6:    $D_{u_{t-1}}(x_t) \leftarrow$ Allocation output for $x_t$
7:    $D(x_t) \leftarrow D_{u_{t-1}}(x_t)/\text{sum}(D_{u_{t-1}}(x_t))$
8:    $D_{comb} \leftarrow \mu \cdot D_{dSim}(x_t) + (1-\mu)\cdot D(x_t)$
9:    $C \leftarrow$ Sample $k$ experts from $D_{comb}$
10:   $\hat{y}_t \leftarrow$ Aggregate label from committee $C$
11:   $S \leftarrow S \cup \{(x_t,y_t)\}$
12:   **if** $|S| = B$ **then**
13:     $u \leftarrow u - \eta \cdot \frac{\partial \mathscr{L}_D(u)}{\partial u}\Big|_S$
14:     $S \leftarrow \emptyset$
15: return $u_T$
```

Theoretical Analysis. Analyzing the two algorithms that use *dSim* shows that the trained allocation model simulates the accuracy functions of the annotators. Our first theorem states that if annotator e_j has high accuracy for category z, then how fast our algorithms converge to a model that assigns high weight to e_j depends on the $dSim(e_j,z)$.

Theorem 2.3. *For any group z, assume annotator e_j is more accurate than all others. For $\beta > 0$, suppose we set dSim function* st. $dSim(e_j,z) - \max_{j' \in \{1,\dots,m\}\setminus\{j\}} dSim(e_{j'},z) \geq \beta$. *Assume all annotators receive the same rewards/penalties for correct/incorrect predictions. Then the training algorithm that initializes $D(x)$ parameters u with this dSim function increases the weight assigned to annotator e_j by at least $2\beta\delta$ in expectation, where $\delta \in [0,1]$ depends on the choice of δ_{reward} and $\delta_{penalty}$ values for the given input.*

Hence, larger the *dSim* weight for e_j, larger is their weight in the final allocation model. Secondly, we show that when using appropriate *dSim*, if there are accurate annotators who are not assigned a high weight by *dSim*, they will be "discovered" during the training.

Theorem 2.4. *For any group z, assume annotator e_j has perfect accuracy. Let k be the size of the committee sampled from $D(x)$ to label input x. Let the dSim function be set such that $dSim(e_j,z) = \varepsilon$, for some $\varepsilon \in [0,1]$, but the total weight (normalized) assigned by dSim to accurate annotators for group z is greater than 0.5. Assume all annotators receive the same rewards for correct prediction and same penalties for incorrect prediction. Then, there is an expected positive increase in the weight of this annotator if $\varepsilon > 1-(1-k/(2m))^{1/k}$.*

Hence, our algorithms can discover accurate annotators so long as other accurate annotators are available to infer the true labels for this input category and k,ε are sufficiently large. The proofs for both theorems are provided in the Supplementary Material.

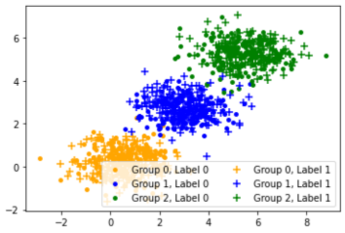

Figure 2. Synthetic clusters in §3.

Method	Label Acc.	Assignment Acc.
Smooth-Matching	.90 (.08)	.87 (0.27)
Strict-Matching	.79 (.01)	.74 (0.36)
Goel and Faltings [15]	.50 (.01)	.33 (.00)
Tran-Thanh et al. [49]	.50 (.01)	.33 (.01)
Keswani et al. [28]	.41 (.09)	.17 (.26)

Table 1. Label and allocation accuracy for §3 with noise s=0.3. We report mean accuracy over 50 trials, with standard error in brackets.

3. Evaluating task allocation on a synthetic dataset

Consider a binary classification task with three annotators having distinct areas of expertise, denoted by the colors orange, blue, and green. Assume each annotator is a perfect oracle when asked to label an example in their respective area but only 20% accurate in the other two areas, exhibiting consistent bias outside their respective areas of expertise. In the best case, the task allocator will correctly assign each input to the correct expert, yielding perfect labeling accuracy. In the worst case, assigning every input to the wrong annotator will yield around 20% accuracy. Because experts are assumed to be perfect oracles, each correct task allocation ensures a correct label. Consequently, task allocation accuracy largely determines label accuracy, which is lower-bounded by allocation accuracy.

As data, we generate 10,000 2D points, each represented by a (x,y) coordinate and drawn from one of three clusters, corresponding to the three areas (colors) of expertise. We begin by sampling μ ∼Unif$[0,1]$ and constructing a 2D diagonal matrix Σ, with diagonal entries sampled from Unif$[0,1]$. Points are then sampled roughly equally from the three clusters as follows: $\mathcal{N}(\mu,\Sigma)$ (orange), $\mathcal{N}(\mu+2.5,\Sigma)$ (blue), and $\mathcal{N}(\mu+5,\Sigma)$ (green). Every point is randomly assigned either label 0 ('•') or 1 ('+'). **Figure** 2 shows the dataset. Because class labels are assigned randomly, a classifier knowing only a point's (x,y) coordinates can only achieve 50% accuracy. Similarly, a task allocator knowing only the (x,y) coordinates has a 1/3 chance of assigning the input to the correct expert.

Specifying ***dSim*****.** Let e_c denote the expert corresponding to color c. For input x, the optimal $dSim(e_c,x)$ would be 1 when x has color c and 0 otherwise, perfectly assigning each example to the appropriate expert. To investigate the effect of varying informativeness of prior information, we introduce noise parameter $s \in [0,2/3]$ and define $dSim(e_c,x)$ as follows: $dSim(e_c,x) = 1-s$ if x has color c and $s/2$ otherwise. With no noise ($s=0$), we revert to the optimal $dSim(e_c,x)$ specified above. Maximal noise ($s=2/3$) yields $\forall_x dSim(e_c,x) = 1/3$: a uniform distribution over all annotators. Additional methodological details are provided in Appendix C in Supplementary Material (see §7).

Baselines. (1) Goel and Faltings [15] learn a task allocation policy using accuracy estimates for all annotators from a history of gold standard tasks. Because we assume that gold standard tasks are unavailable, we instead run their algorithm with accuracy estimates derived using noisy, aggregated annotator predictions. (2) Tran-Thanh et al. [49] learn an allocation using a multi-arm bandit approach, with initial exploration steps to estimate annotator accuracies followed by exploitation steps that assign inputs to annotators using estimated accuracies. Once again, in absence of gold standard, the accuracy estimates in the exploration step of their algorithm are obtained using aggregated predictions from the annotators. (3) Keswani et al. [28]'s method is equivalent to training an input-specific task allocation model using Algorithm Smooth-Matching without any prior information, i.e., without *dSim*. In this case, the training algorithm will start with a random allocation. Lacking prior information, we expect all three of these baselines to struggle in the closed-training setting we consider. See Appendix C for further implementation details.

Results. We first assume moderate noise $s = 0.3$, roughly the middle of $s \in [0, 2/3]$. **Table 1** compares Strict-Matching and Smooth-Matching vs. baseline algorithms [15, 49, 28]. We observe large differences in mean label accuracy: 0.90, 0.79, 0.50, 0.50, and 0.41, respectively. Standard error over 50 trials shows that differences are statistically significant. As noted earlier, our assumption of oracle experts means that task allocation accuracy largely determines label accuracy, as the task allocation results here confirm: 0.87, 0.74, 0.33, 0.33, and 0.17, respectively. The vast difference in task allocation accuracy in training with *dSim* (Smooth-Matching, Strict-Matching) and without *dSim* (Keswani et al. [28] baseline) shows the critical importance of prior information in training. The weaker accuracy of Strict-Matching vs. Smooth-Matching can be explained by their differing use of *dSim*. Whereas Smooth-Matching exploits *dSim* for initialization only, Smooth-Matching continues to benefit from *dSim* by utilizing it throughout training.

Figure 1 in Appendix C shows performance for varying s values. As expected, increasing values for noise s leads to a corresponding decrease in accuracy, and for large values of s, *dSim* degrades toward the uniform distribution as an uninformative prior.

4. Evaluating task allocation on a real-world dataset: toxicity detection

Dataset. Civil Comments [7] provides toxicity labels for 1.8M news comments. Of these, 450K comments are also labeled for the demographic group targeted (e.g., LGBTQ+, race, etc.). We consider the binary classification task of predicting whether a comment contains an "identity attack": a comment that is toxic and targets a specific demographic affiliation.

Goyal et al. [19] augment Civil Comments with additional demographic identity labels of the annotators. This enables study of how annotator identity may influence their toxicity ratings. They sample 25.5k comments to augment with additional labels, uniformly sampling comments from three targeted groups: LGBTQ, African-American, and Control (identity agnostic). Of these, 12% of comments are labelled as containing an identity attack. Roughly 1K crowd annotators contributed labels to their study, with around 1/3 of annotators affiliated with each demographic group. Each comment is labeled by 5 annotators from each group (i.e., 15 in total). See Appendix D for additional details.

Control annotators often label toxicity differently than annotators whose own demographic group is targeted in a comment. **Table 1** in the Appendix shows illustrative examples. Such differences in toxicity ratings by annotator demographic indicate a form of consistent bias, motivating our consideration of demographics in task allocation.

Specifying *dSim*. We investigate potential allocation accuracy improvement by matching annotator demographics to the target groups. For comment x that targets demographic group g and annotator e, we define $dSim(z, e) = 1$, if e identifies with g and 0 otherwise.

Baselines. We evaluate against baseline training algorithms from Goel and Faltings [15], Tran-Thanh et al. [49] and Keswani et al. [28]. The descriptions of these baselines are provided in §3. See Appendix D in Supplementary Material (see §7) for model and implementation details of our algorithms and the baselines.

Measurement. We follow Goyal et al. [19] in reporting AUC score: the area under the receiver operating characteristic (ROC) curve. The dataset is skewed (only 12% of comments contain identity attacks), and AUC is appropriately sensitive to such class imbalance. We randomly split the dataset into train-test partitions (70-30 split), evaluate methods across 25 trials of different splits, and report AUC mean and standard error.

Alternative Gold Standards. We measure label accuracy using two views of ground truth: 1) the classic assumption of a single, objective gold standard vs. 2) that the gold standard is subjective and varies by community [29, 17, 19]. For the objective gold setting, we induce gold by majority vote over all annotators from Civil Comments [7]. Note that these annotators used to define the objective gold are completely disjoint from the set of annotators available for our task allocation experiments. In the subjective setting, we induce gold using the majority vote over the 5 annotators in [19] whose demographic identity matches the comment's target demographic. Here, the annotators available for task allocation include the experts whose majority vote determines gold. Again, gold labels are never used for training allocation models, but for evaluation purposes only.

Results. Table 2 present results for objective and subjective gold conditions. In both settings, training without prior information yields lower accuracy. We observe improvement in performance when using prior information despite the fact that differences in annotator accuracies across demographics are not statistically-significant. This is because, after training, allocation weights for each input contain information from both prior and observed samples, and correspondingly every test input is assigned to the top-ranked annotator for that input. The accuracy of the top-ranked annotator is often better than the average annotator accuracy, leading to improved prediction scores.

Table 2.: Objective and subjective gold results. We report AUC over 25 trials (standard error in brackets).

Method	Objective gold AUC Score	Subjective gold AUC Score
Smooth-Matching	.62 (.01)	.71 (.01)
Strict-Matching	.59 (0)	.67 (.01)
Goel and Faltings [15]	.60 (.01)	.64 (.02)
Tran-Thanh et al. [49]	.60 (0)	.66 (.01)
Keswani et al. [28]	.60 (.01)	.66 (.01)

Results: objective gold. We observe negligible standard error (~ 0.01) in AUC scores across trials, indicating consistency of the mean AUC scores for comparing methods. Smooth-Matching achieves the best AUC score (0.62), 2% better than prior work baselines that lack prior information (i.e., training without *dSim*). Strict-Matching performs 1% worse than these baselines, likely due to insufficiency of using *dSim* only for initialization. In contrast, Smooth-Matching mitigates this issue by using *dSim* throughout the training.
Results: subjective gold. Smooth-Matching again achieves the top mean AUC score (0.71), with 5-7% improvement over baselines. In contrast with the objective gold setting, Strict-Matching also outperforms all baselines (1-3%). In general, we observe both larger margins and higher overall scores than in the objective gold setting. In part, this may reflect a simple dataset artifact (e.g., all methods perform better in the subjective vs. objective gold setting). We also noted a minor artifact earlier in experimental design that could inflate scores here: whereas the objective gold setting uses disjoint annotator pools to define gold vs. task allocation, here the annotator pool for task allocation also includes the 5 annotators who define the community gold standard. Despite these confounds, strong intuition remains to expect greater benefit from task allocation in the community-gold setting: when community members are empowered to define gold for their community, we stand to benefit more from engaging them as annotators for their community.

5. Discussion and limitations

Availability of *gold standard*. As mentioned in §2.1, prior studies on task allocation often assume that annotator correctness can be accurately determined. However, in real life, we only have access to fallible human decisions and task allocation seeks to find the suitable annotators whose aggregated decision is considered the gold [25]. Given that humans define the gold, the assumption that we can accurately determine annotator correctness is not always true. That is not to say correctness can never be determined; for tasks with objectively correct answers, e.g. question-answer tasks [13], annotator qualities can be measured. But when the ground truth is subjective (e.g., in toxicity analysis) or for settings that are yet unexplored through crowdsourcing (e.g., regional language moderation), assuming presence of gold labels can be unrealistic. The subjectivity of ground truth is an important factor. The predominant view in crowdsourcing, that the *gold standard* is the majority decision of a random group of annotators, has been challenged by many studies and community-defined gold standards have thus been forwarded as a way to incorporate minority voices [19, 29, 41, 17, 44]. Our framework, hence, takes a contextual approach to task allocation. Providing annotator background as prior information ensures that the social context of the tasks is taken into account. Training our closed-loop framework does not require any ground truth, ensuring separation from predefined ideas of "correctness".
Prior information and *dSim*. Through empirical evaluations, we show that prior information can improve closed-loop model training. However, certain settings may not require such prior information for accurate training, e.g. tasks where the ground truth is considered objective (e.g., factual question-answer datasets) or when annotator qualities are not input-specific. Secondly, providing incorrect or non-contextual prior information to the framework can have a negative impact on the training process. An incorrect estimate of *dSim* can lead to incorrect allocation in the initial iterations, thus derailing the entire training process (as observed for noisy *dSim* in § 3). Finally, a key assumption we make is that annotator demographics and target demographics are known. While it is reasonable to expect annotator demographics to be provided (e.g., using in-take surveys during onboarding), demographics associated with the tasks (e.g., groups targeted in social media posts) may not always be available. In practice, target demographic would have to be manually labeled or automatically detected (with noise). Tackling noise in target demographics merits future exploration and can improve our framework's applicability.
Annotator consultation costs. Different annotators can have different consultation costs. E.g., platforms like Upwork [20] allow clients to employ human experts (or freelancers) for their posted jobs. Experts often have more experience/training (and higher prices) than generalist workers. Our framework supports each person having such additional input-specific costs. Let $c_{e_j} : \mathscr{X} \to \mathbb{R}$ for e_j denote the input-specific cost function for annotator e_j. To incorporate input-specific costs, we can alternately minimize a regularized loss function: $\mathscr{L} := \mathscr{L}_D + \lambda \cdot \mathbb{E}_x \left[D(x)^\top c(x)\right]$, where $\lambda \geq 0$ is the cost hyperparameter. Minimizing this cost-regularized loss will ensure that the annotator costs are accounted.
Updating annotators. To add a new annotator, we can assign them weight proportional to their *dSim* value for any given input category and subsequent training will update this weight based on their predictions. Removing an annotator, however, can affect performance if this annotator had expertise in subspaces where all other annotators are inaccurate. If the removed annotator did not have any unique expertise, then choosing a large committee size can partially ameliorate this issue. However, alternate methods to deal with annotator removal would be beneficial in practice and merit further exploration.

6. Related Work

Multiple studies in crowdsourcing have evaluated the importance of repeated labeling and proposed methods to handle annotator heterogeneity [45, 25, 31, 58, 53, 27, 6, 15]. Traditionally, most studies consider heterogeneity amongst annotators but not amongst the input tasks. In contrast, our framework constructs input-specific allocation models. Certain recent papers propose methods to handle correlated heterogeneity of annotators and tasks in offline settings, where all annotators provide a decision for most tasks and the goal is to infer ground truth from given annotations [57, 48, 9]. However, offline settings can be expensive as it requires a large number of annotations. We tackle online settings where the relevant experts are chosen judiciously to obtain relatively larger cost-benefit.

Studies on online allocation forward a variety of methods to construct appropriate allocation models. Yin et al. [56] design budget-limited allocation policies that match the annotator preferences to task requirements. However, annotator preferences can be unavailable and would be susceptible to implicit biases. Liu et al. [32], Li et al. [30] propose frameworks that learn annotator accuracies by comparing to ground truth for completed tasks. §2.1 shows that such frameworks perform poorly in closed-loop settings; they can potentially be employed if our proposed approach of using prior information is incorporated into their designs. Fan et al. [13] develop allocation models that assign tasks to annotators who have past experience with similar tasks. However, estimating inter-task similarity can be expensive when size and variability across tasks are large. Certain studies estimate annotator cognitive abilities or use their social network profiles to allocate tasks appropriately [22, 16, 12]. In absence of subsequent training, these methods will have low accuracy when the annotator profiles have insufficient information about their qualities. Ho et al. [23] and Ho and Vaughan [24] study a related but different allocation setup where human annotators' skill levels are known in advance. Bandit approaches [2, 33, 52, 49] can also be used for allocation but primarily assume access to true rewards/penalties. In closed-loop settings, they can suffer from exacerbated inaccuracies, as observed in §3.

Recent human-in-the-loop research studies deferral frameworks that train an automated classifier which can either make a prediction or defer the decision to a human [28, 36, 34]. Deciding whether a prediction should be made by the classifier or (one or more) humans is a task allocation problem. However, here again prior algorithms for deferral training assume access to true class labels for training [28, 36, 34, 21], which are unavailable in our closed-loop setting. In case of limited ground truth information, semi-supervised classification [8, 50, 37, 55, 26] selectively use either the model's prediction or labels from noisy crowd annotators to appropriately re-train the model. While the goal of these approaches is to train a classifier, our primary goal is to train an input-specific task allocation model that learns every human decision-maker's error region.

7. Conclusion

We initiate a study of a closed-loop online task allocation framework where decisions from the human annotators are used to continuously train the allocation model as well. We provide algorithms that utilize the available prior information about the annotators to bootstrap an accurate training process. By encoding prior information about the human annotators, e.g. demographics and background, we ensure that the learned allocation models are contextually-relevant. **Appendix is provided as supplementary material available at** `https://tinyurl.com/22e9zpuu` **and the code is available at** `https://github.com/vijaykeswani/Closed-Loop-Task-Allocation`.

References

[1] Anagnostopoulos A, Becchetti L, Castillo C, Gionis A, Leonardi S. Online team formation in social networks. In: Proceedings of the 21st international conference on World Wide Web; 2012. p. 839–848.

[2] Arora S, Hazan E, Kale S. The multiplicative weights update method: A meta-algorithm and applications. Theory of Computing 2012;8(1):121–164.

[3] Awerbuch B, Kleinberg R. Online linear optimization and adaptive routing. Journal of Computer and System Sciences 2008;74(1):97–114.

[4] Bertrand M, Mullainathan S. Are Emily and Greg more employable than Lakisha and Jamal? A field experiment on labor market discrimination. American economic review 2004;94(4):991–1013.

[5] Bhuiyan MM, Zhang AX, Sehat CM, Mitra T. Investigating differences in crowd-sourced news credibility assessment: Raters, tasks, and expert criteria. Proceedings of the ACM on Human-Computer Interaction 2020;4(CSCW2):1–26.

[6] Bonald T, Combes R. A minimax optimal algorithm for crowdsourcing. Advances in Neural Information Processing Systems 2017;30.

[7] Borkan D, Dixon L, Sorensen J, Thain N, Vasserman L. Nuanced metrics for measuring unintended bias with real data for text classification. In: Companion proceedings of the 2019 world wide web conference; 2019. p. 491–500. Dataset:`https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data`.

[8] Chapelle O, Schölkopf B, Zien A. Semi-supervised Learning. MIT Press; 2010.

[9] Davani AM, Díaz M, Prabhakaran V. Dealing with disagreements: Looking beyond the majority vote in subjective annotations. Transactions of the Association for Computational Linguistics 2022;10:92–110.

[10] Davidson T, Bhattacharya D, Weber I. Racial Bias in Hate Speech and Abusive Language Detection Datasets. In: Proceedings of the Workshop on Abusive Language Online; 2019. .

[11] Dawid AP, Skene AM. Maximum likelihood estimation of observer error-rates using the EM algorithm. Journal of the Royal Statistical Society: Series C (Applied Statistics) 1979;28(1):20–28.

[12] Difallah DE, Demartini G, Cudré-Mauroux P. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In: Proceedings of the 22nd international conference on World Wide Web; 2013. p. 367–374.

[13] Fan J, Li G, Ooi BC, Tan Kl, Feng J. icrowd: An adaptive crowdsourcing framework. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data; 2015. p. 1015–1030.

[14] Fontenla-Romero Ó, Guijarro-Berdiñas B, Martinez-Rego D, Pérez-Sánchez B, Peteiro-Barral D. Online machine learning. In: Efficiency and Scalability Methods for Computational Intellect IGI Global; 2013.p. 27–54.

[15] Goel N, Faltings B. Crowdsourcing with fairness, diversity and budget constraints. In: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society; 2019. .

[16] Goncalves J, Feldman M, Hu S, Kostakos V, Bernstein A. Task routing and assignment in crowdsourcing based on cognitive abilities. In: Proceedings of the 26th International Conference on World Wide Web Companion; 2017. p. 1023–1031.

[17] Gordon ML, Lam MS, Park JS, Patel K, Hancock J, Hashimoto T, et al. Jury learning: Integrating dissenting voices into machine learning models. In: CHI Conference on Human Factors in Computing Systems; 2022. p. 1–19.

[18] Gorwa R, Binns R, Katzenbach C. Algorithmic content moderation: Technical and political challenges in the automation of platform governance. Big Data & Society 2020;7(1):2053951719897945.

[19] Goyal N, Kivlichan I, Rosen R, Vasserman L. Is Your Toxicity My Toxicity? Exploring the Impact of Rater Identity on Toxicity Annotation. The 26th ACM Conference On Computer-Supported Cooperative Work And Social Computing (CSCW) 2022;Dataset:https://www.kaggle.com/datasets/google/jigsaw-specialized-rater-pools-dataset.

[20] Green DD, et al. Fueling the gig economy: a case study evaluation of Upwork. com. Manag Econ Res J 2018;4(2018):3399.

[21] Hemmer P, Schellhammer S, Vössing M, Jakubik J, Satzger G. Forming Effective Human-AI Teams: Building Machine Learning Models that Complement the Capabilities of Multiple Experts. arXiv preprint arXiv:220607948 2022;.

[22] Hettiachchi D, Van Berkel N, Kostakos V, Goncalves J. CrowdCog: A Cognitive skill based system for heterogeneous task assignment and recommendation in crowdsourcing. Proceedings of the ACM on Human-Computer Interaction 2020;4(CSCW2):1–22.

[23] Ho CJ, Jabbari S, Vaughan JW. Adaptive Task Assignment for Crowdsourced Classification. In: International Conference on Machine Learning; 2013. p. 534–542.

[24] Ho CJ, Vaughan J. Online task assignment in crowdsourcing markets. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 26; 2012. p. 45–51.

[25] Ipeirotis PG, Provost F, Wang J. Quality management on amazon mechanical turk. In: Proceedings of the ACM SIGKDD workshop on human computation; 2010. p. 64–67.

[26] Kajino H, Tsuboi Y, Kashima H. A convex formulation for learning from crowds. In: Twenty-Sixth AAAI Conference on Artificial Intelligence; 2012. .

[27] Karger DR, Oh S, Shah D. Budget-optimal task allocation for reliable crowdsourcing systems. Operations Research 2014;62(1):1–24.

[28] Keswani V, Lease M, Kenthapadi K. Towards Unbiased and Accurate Deferral to Multiple Experts. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society; 2021. .

[29] Kumar D, Kelley PG, Consolvo S, Mason J, Bursztein E, Durumeric Z, et al. Designing toxic content classification for a diversity of perspectives. In: Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021); 2021. p. 299–318.

[30] Li H, Zhao B, Fuxman A. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In: Proceedings of the 23rd international conference on World wide web; 2014. p. 165–176.

[31] Liu Q, Peng J, Ihler AT. Variational inference for crowdsourcing. Advances in neural information processing systems 2012;25.

[32] Liu X, Lu M, Ooi BC, Shen Y, Wu S, Zhang M. CDAS: A Crowdsourcing Data Analytics System. Proceedings of the VLDB Endowment 2012;5(10).

[33] Lu T, Pál D, Pál M. Contextual multi-armed bandits. In: Proceedings of the Thirteenth international conference on Artificial Intelligence and Statistics JMLR Workshop and Conference Proceedings; 2010. p. 485–492.

[34] Madras D, Creager E, Pitassi T, Zemel R. Learning adversarially fair and transferable representations. arXiv preprint arXiv:180206309 2018;.

[35] Mehrabi N, Morstatter F, Saxena N, Lerman K, Galstyan A. A survey on bias and fairness in machine learning. arXiv preprint arXiv:190809635 2019;.

[36] Mozannar H, Sontag D. Consistent estimators for learning to defer to an expert. In: International Conference on Machine Learning PMLR; 2020. p. 7076–7087.

[37] Nguyen AT, Wallace BC, Lease M. Combining crowd and expert labels using decision theoretic active learning. In: Third AAAI conference on human computation and crowdsourcing; 2015. .

[38] Pan Y, Froese F, Liu N, Hu Y, Ye M. The adoption of artificial intelligence in employee recruitment: The influence of contextual factors. The International Journal of Human Resource Management 2021;p. 1–23.

[39] Raghu M, Blumer K, Sayres R, Obermeyer Z, Kleinberg B, Mullainathan S, et al. Direct uncertainty prediction for medical second opinions. In: International Conference on Machine Learning; 2019. p. 5281–5290.

[40] Sap M, Card D, Gabriel S, Choi Y, Smith NA. The risk of racial bias in hate speech detection. In: Proceedings of ACL; 2019. p. 1668–1678.

[41] Sap M, Swayamdipta S, Vianna L, Zhou X, Choi Y, Smith NA. Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. NAACL 2022;.

[42] Schumann C, Foster J, Mattei N, Dickerson J. We need fairness and explainability in algorithmic hiring. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS); 2020. .

[43] Schütze H, Manning CD, Raghavan P. Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge; 2008.

[44] Sen S, Giesel ME, Gold R, Hillmann B, Lesicko M, Naden S, et al. Turkers, scholars," arafat" and" peace" cultural communities and algorithmic gold standards. In: Proceedings of the 18th acm conference on computer supported cooperative work & social computing; 2015. p. 826–838.

[45] Sheng VS, Provost F, Ipeirotis PG. Get another label? improving data quality and data mining using multiple, noisy labelers. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining; 2008. p. 614–622.

[46] Spinde T, Rudnitckaia L, Sinha K, Hamborg F, Gipp B, Donnay K. MBIC–A Media Bias Annotation Dataset Including Annotator Characteristics. iConference 2021;.

[47] Surowiecki J. The wisdom of crowds. Anchor; 2005.

[48] Tao F, Jiang L, Li C. Label similarity-based weighted soft majority voting and pairing for crowdsourcing. Knowledge and Information Systems 2020;62(7):2521–2538.

[49] Tran-Thanh L, Stein S, Rogers A, Jennings NR. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. Artificial Intelligence 2014;214:89–111.

[50] Triguero I, García S, Herrera F. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowledge and Information systems 2015;42(2):245–284.

[51] Valcarce D, Parapar J, Barreiro Á. Additive smoothing for relevance-based language modelling of recommender systems. In: Proceedings of the 4th Spanish Conference on Information Retrieval; 2016. p. 1–8.

[52] Valera I, Singla A, Gomez Rodriguez M. Enhancing the Accuracy and Fairness of Human Decision Making. Advances in Neural Information Processing Systems 2018;31:1769–1778.

[53] Welinder P, Branson S, Perona P, Belongie S. The multidimensional wisdom of crowds. Advances in neural information processing systems 2010;23.

[54] Wu G, Chen Z, Liu J, Han D, Qiao B. Task assignment for social-oriented crowdsourcing. Frontiers of Computer Science 2021;15(2):1–11.

[55] Yan Y, Rosales R, Fung G, Dy JG. Active learning from crowds. In: International Conference of Machine Learning; 2011. .

[56] Yin X, Chen Y, Xu C, Yu S, Li B. Matchmaker: Stable Task Assignment With Bounded Constraints for Crowdsourcing Platforms. IEEE Internet of Things Journal 2020;8(3).

[57] Zhang H, Jiang L, Xu W. Multiple Noisy Label Distribution Propagation for Crowdsourcing. In: IJCAI; 2019. p. 1473–1479.

[58] Zhou D, Basu S, Mao Y, Platt J. Learning from the wisdom of crowds by minimax entropy. Advances in neural information processing systems 2012;25.