The Disagreement Deconvolution: Bringing Machine Learning Performance Metrics In Line With Reality

Mitchell L. Gordon
Stanford University
mgord@cs.stanford.edu

Kaitlyn Zhou
Stanford University
katezhou@stanford.edu

Kayur Patel
Apple Inc.
kayur@apple.com

Tatsunori Hashimoto
Stanford University
thashim@stanford.edu

Michael S. Bernstein
Stanford University
msb@cs.stanford.edu

ABSTRACT

Machine learning classifiers for human-facing tasks such as comment toxicity and misinformation often score highly on metrics such as ROC AUC but are received poorly in practice. Why this gap? Today, metrics such as ROC AUC, precision, and recall are used to measure technical performance; however, human-computer interaction observes that evaluation of human-facing systems should account for people's reactions to the system. In this paper, we introduce a transformation that more closely aligns machine learning classification metrics with the values and methods of user-facing performance measures. The disagreement deconvolution takes in any multi-annotator (e.g., crowdsourced) dataset, disentangles stable opinions from noise by estimating intra-annotator consistency, and compares each test set prediction to the individual stable opinions from each annotator. Applying the disagreement deconvolution to existing social computing datasets, we find that current metrics dramatically overstate the performance of many human-facing machine learning tasks: for example, performance on a comment toxicity task is corrected from .95 to .73 ROC AUC.

CCS CONCEPTS

• **Human-centered computing → HCI design and evaluation methods**; **Collaborative and social computing design and evaluation methods**; • **Computing methodologies → Machine learning**.

ACM Reference Format:
Mitchell L. Gordon, Kaitlyn Zhou, Kayur Patel, Tatsunori Hashimoto, and Michael S. Bernstein. 2021. The Disagreement Deconvolution: Bringing Machine Learning Performance Metrics In Line With Reality. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3411764.3445423

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445423

1 INTRODUCTION

The machine learning classifiers that underpin modern social computing systems such as Facebook, Wikipedia, and Twitter are a study in contrasts. On one hand, current performance metrics for popular tasks such as comment toxicity classification and disinformation detection are extremely high, featuring up to .95 ROC AUC [11, 72], making the problem appear nearly solved. On the other hand, audits suggest that these algorithms' performance is in reality much poorer than advertised [56]. Even well-intentioned platforms will continue to make highly-publicized mistakes [16] until they can better align metrics with reality.

This disconnect between classifier performance and user-facing performance is indicative of a larger disconnect between how machine learning (ML) and human-computer interaction (HCI) researchers evaluate their work. ML aims to evaluate technical performance, developing metrics that measure generalization error over unseen examples such as precision, recall, and ROC AUC. HCI instead aims to report user-facing experience, developing metrics that measure direct user response or opinion such as agreement rates, Likert scales, and behavioral outcomes. In many cases, common metrics used for technical performance are already aligned with user-facing performance, as strong gesture recognition [70] or text entry prediction rates [64] also represent strong user feedback. However, in the common ML metrics used for several critical tasks in social computing and HCI scenarios such as comment toxicity, the labels used to evaluate generalization error often do not directly map to user responses or opinions. As a result, strong technical performance over common metrics appears substantially and falsely inflated above user-facing performance.

In this paper, we introduce a transformation that more closely aligns ML classification metrics with the values and methods of user-facing performance measures. We base our transformation on the observation that common technical performance metrics flatten multiple opinions into a single ground truth label for each example, incorrectly ignoring disagreement between people [49, 63]. For an HCI researcher or professional who is measuring perceptions of toxicity, the quantity of interest is rarely a single bit of *whether* a given comment is toxic. At the very least, they'd like to know *the proportion of people* who view the comment as toxic. The ML practice of aggregating multiple annotators' responses for each example via majority vote or other approaches [61] into a single ground truth label [2] makes sense when the whole population tends to agree on the answer to the question, such as "Does this image contain a cat?" in object classification or "Does the word

'java' in this sentence refer to coffee or the programming language?" in word sense disambiguation. However, social computing tasks and other common tasks in HCI often remain deeply contested, with much lower levels of agreement amongst the population [12]. Our goal is to incorporate an estimate of how contested each label will be into ML metrics in an interpretable way, producing metrics that ask not "What proportion of ground truth labels does the classifier agree with?" but "What proportion of the population does the classifier agree with?" For traditional ML tasks such as object classification, the resulting metrics would likely be very similar; but in social computing tasks and many others in HCI, the measurements could diverge substantially.

Prior work has yet to develop a metric that achieves this goal. ML researchers have begun training models to output a distribution of labels and then evaluating that distribution, such as cross-entropy compared to the distribution of annotators' labels [44, 52, 53, 68, 71]. While cross-entropy acknowledges the existence of disagreement, it doesn't measure performance in a real-world setting because in practice models for many social computing and HCI classification tasks often must make a single discrete decision [12]. This means that a model that exactly reproduces the distribution can achieve perfect technical performance but face poor user-facing performance: YouTube's demonetization pipeline must ultimately decide whether to demonetize a video; a commenting platform must decide whether to remove a comment; a social network must decide whether to place a disinformation flag on a post. We therefore require a metric that asks: given a classifier can only make one decision, what would each annotator think of it? Unfortunately, developing a metric that accounts for levels of disagreement directly remains challenging because of label uncertainty [36]: even well-intentioned labelers can make accidental errors, misunderstand questions, or even disagree with themselves later when asked the same question a second time [14]. These challenges mean that the observed distribution of labels may not reflect the true underlying distribution of beliefs. Developing a metric that represents the proportion of the population that would agree with each classification decision requires a method that more precisely estimates that proportion while factoring out possible errors.

Our method, the *disagreement deconvolution*, transforms any ML classification metric to reflect the underlying distribution of population labels. We set our goal as identifying whether each label represents the annotator's *primary label*—the label they would provide the most often in an imagined scenario where they were to label the item repeatedly over time—or intuitively, their dominant response to the item when setting aside any errors, mistakes, or occasional inconsistencies. Our technique separates (deconvolves) the observed set of labels into two sets: (1) labels that do not represent primary labels, which might arise due to accident and error, and (2) primary labels, which we use as the basis for evaluation metric. In other words, the disagreement deconvolution first estimates an idealized annotation process where each annotator always provides their primary label. Then, when computing an evaluation metric such as precision or ROC AUC, instead of comparing each prediction to a single "ground truth", we compare each prediction to multiple different "ground truths", once for each annotator's primary label. We derive this deconvolution by estimating p_{flip}, the probability that a given annotation will differ from the annotator's primary label. We demonstrate that p_{flip} can be directly estimated from many existing crowdsourcing datasets, where a subset of questions were asked multiple times as attention checks [34]. For datasets where p_{flip} is not directly measurable, we derive an approximation based on the singular value decomposition of the item-annotation matrix. The output of the disagreement deconvolution is a version of the test set that comprises a distribution of annotators' primary labels. Any performance metric (e.g., precision, recall) can then sample from this distribution to answer the question, "What distribution of the population agrees with this classification?"

We apply the disagreement deconvolution to evaluate a set of popular social computing tasks, finding that current metrics dramatically overstate the capabilities of classifiers for social computing tasks. For instance, in the popular Jigsaw toxicity task where Jigsaw claims that models achieve .95 ROC AUC, we find that even an oracle classifier that perfectly predicts each aggregated annotation could only achieve a disagreement-adjusted ROC AUC of .73. We verify that applying the disagreement deconvolution to a sample of classic ML tasks, like object recognition, results in little adjustment, reinforcing that while today's standard aggregated evaluation approach is indeed appropriate for classic ML tasks, it is unsuited to social computing tasks.

Given our results, it seems likely that design, product, and research teams are at risk of training and launching classifiers that they believe to be highly accurate but in practice are not. We envision that the disagreement deconvolution could be used to help teams take a more human-centered view of the classifiers they are creating. Instead of relying on a single aggregated pseudo-human, this approach enables them to better understand how a collection of individuals would respond to the decisions their classifiers make. Using the disagreement deconvolution, our work also defines a new oracle evaluation measure that estimates the best achievable disagreement-adjusted performance by *any* classifier. We show that this oracle evaluation can be computed as a diagnostic prior to collecting a full dataset, which allows teams to use this diagnostic to guide task and ultimately product design. Teams can either iterate on their task until they determine that the remaining disagreement cannot be reasonably resolved (which may still produce poor performance) and/or design downstream methods to deal with contested situations.

2 RELATED WORK

In this section, we motivate the disagreement deconvolution through an integration of the social computing, crowdsourcing, human-computer interaction, and machine learning research literatures.

2.1 Classifiers and disagreement in social computing problems

Across tasks such as identifying toxic comments [31, 66], bot accounts [67], and misinformation [72], researchers and platforms [40] increasingly turn to ML to aid their efforts [32]. Specifically, these models are often trained using a supervised learning pipeline where we:

(1) Collect a large dataset of individual beliefs, either generated through crowdsourcing services that ask several labelers to

annotate each item according to policy and then aggregate the result into a single ground truth label (e.g., [21]), or similarly by asking and then aggregating experts (e.g., [65]).
(2) Use those ground truth labels to train a model that produces either a discrete binary prediction or a continuous probabilistic prediction for any given example.
(3) Evaluate the resulting model by comparing predictions to the ground truth labels in a held-out test set.

For instance, in a Kaggle competition that received over 3,000 submissions, researchers were challenged to discover the best-performing architecture in a toxicity detection task [38]. Facebook makes the vast majority of moderation decisions through classifiers [8]. YouTube creates classifiers that decide which videos need to be demonetized [15].

While modern deep learning techniques that rely on this pipeline now outperform human judgment on many artificial intelligence (AI) tasks such as image classification [21], the results for social computing tasks remain problematic. Despite ostensibly strong reported test scores on these tasks [38, 62], the algorithms are roundly criticized for making obvious mistakes [1]. Recent efforts seek to reduce bias [48]; make models more robust to novel situations [29]; help models better interpret culturally situated language and behavior [30]. Yet there remains a stream of highly publicized mistakes made by these models [1].

One potential explanation for these mistakes may be inherent disagreement within the datasets they are trained on. While a large body of work has established that disagreement exists and investigated its potential sources, less is known about the magnitude of the limitations that this disagreement places on classifiers for social computing tasks.

For instance, interviews with YouTube celebrities, journalists, scientists, politicians, and activists identified very different ideas about what behavior each interviewee felt constituted harassment [46]. It's not clear that simply providing a definition would improve agreement: a study of hate speech annotation found that providing annotators a definition of hate speech did little to improve their agreement [57], with the authors claiming that a "clearer definition is necessary in order to be able to train a reliable classifier" (but did not identify what such a definition might look like). A separate study of hate speech detection labeled 24k tweets using CrowdFlower and reported that while the intercoder-agreement score provided by CF was 92%, only 5% of tweets were coded as hate speech by the majority of coders and only 1.3% were coded unanimously, indicating that the vast majority of tweets that might be hate speech are contested. To create the popular CREDBANK dataset, annotators were asked whether real-world events occurred or not and while 76.54% of events had more than 70% agreement, only 2% of events had 100% agreement among annotators [50].

There are many potential sources of this disagreement: hate speech definitions vary significantly by country [58], and feminists' annotations differ from antiracism activists' annotations which both differ from amateur annotations [69]. Expert moderators of a popular Reddit subreddit often disagree with each other, finding a Fleiss' kappa for comments annotated by two moderators of 0.46, showing only modest agreement between moderators. Further, moderators disagreed with each other on 29% of removed posts [45]. Inter-rater agreement is significantly lower for women than for men in a toxicity task, both under .5 Krippendorph's alpha [10]. Together, this work tells us that there's significant disagreement in these tasks and that its often skewed towards the positive, minority class. However, it remains unclear what level of performance we can expect from models when creating classifiers for these tasks.

In this paper, we establish that this disagreement places fundamental limits on applying today's widely adopted supervised learning pipeline to these problems. We find that social computing tasks cannot possibly achieve performance remotely comparable to that of classic tasks like image detection. We demonstrate this by computing disagreement-adjusted scores using an oracle classifier, which achieves perfect scores on a standard aggregated test set. For classic tasks, the disagreement-adjusted scores remain near perfect, but for social computing tasks, these scores drop dramatically. We show how these fundamental limits are inherently hidden by following the very pipeline that causes them.

Unlike prior work, which focuses its critique on the ML architecture [62], we derive our claim by measuring properties of the *datasets*. We gather a series of datasets for diverse social computing classification tasks such as toxicity detection and misinformation classification, then disaggregate the data into individual labelers' labels for each datapoint.

2.2 Approaches for collecting datasets with disagreement

At the dataset collection stage, crowdsourcing researchers have proposed a number of methods that aim to resolve annotator disagreement either by making task designs clearer or relying on annotators to resolve disagreement among themselves [13, 17, 23, 47, 60]. Other work aims to accurately capture the distribution of annotator opinions [20, 23–25, 41]. Improvements in task design may also come from a new specialization within ML that aims to develop methodologies for data annotation for sociocultural data, drawing from the work of archive scholars [30].

We first note that, in the social computing tasks that we investigate, much of the disagreement is likely irreducible, stemming from the socially contested nature of questions such as "What does, and doesn't, cross the line into harassment?" amongst the population and experts. The above methods may help resolve some disagreement in these datasets, but until such unlikely time as there is ever to be a global consensus on questions such as what constitutes harassment, we demonstrate that even an oracle classifier will be disagreed with.

We therefore envision the disagreement deconvolution as a complementary tool to methods aiming to improve the quality of datasets. First, when used as a diagnostic, it can demonstrate the need for such work in a dataset by showing the maximum classification performance currently possible when the model is deployed, given the level of disagreement in the dataset. Second, it can be used to help track progress as researchers iterate on their datasets. However, we note that methods aiming to resolve disagreement at annotation time could potentially be harmful to the disagreement deconvolution's accuracy were they to be overly aggressive, in which case even the dataset's raw labels may have

collapsed or hidden disagreement, rendering it more challenging for the disagreement deconvolution to recover primary labels.

Our work relates to psychometric and survey design research which has long been interested in test-retest, or reinterview, as a foundation to understand the reliability of both qualitative items, such as achievement tests, attitude questionnaires, and public opinion surveys, and quantitative items, such as most questions found on a census [7, 28, 39, 54]. Seminal work in 1946 established that while a single annotation is sufficient to establish upper and lower bounds for a group's level of agreement on quantitative data, test-retest must be used to establish an upper bound for qualitative data [34]. At a high level, our approach builds off of this idea and uses test-retest annotations to understand a group's distributions of primary labels.

2.3 Evaluating models in the presence of disagreement

2.3.1 Soft labels. When faced with uncertainty, researchers and engineers often rely on models that predict probabilistic labels, representing the model's predicted likelihood that an example belongs to each class. Typically, these models are trained on a single label for each example and output probabilistic labels that might (but do not necessarily) represent disagreement between annotators. Increasingly, researchers argue that these predictions should further embrace disagreement – rather than training on a single label per example, they argue that models should explicitly train on soft labels that represent the distribution of annotations found during the annotation process, and that their predictions should match these soft labels [44, 52, 53, 68, 71]. We agree that it is reasonable to argue that a model that perfectly predicts such distributions has reached its maximum technical performance at training time.

In practice, however, models for many social computing and HCI classification tasks often must make a single discrete decision. If the model is not itself acting as a discrete classifier, then a human viewing that model's output would act as a discrete classifier. For example, YouTube's demonetization pipeline must ultimately decide whether to demonetize a video; a commenting platform must decide whether to remove a comment; a social network must decide whether to place a disinformation flag on a post. So even a model that can perfectly predict the distribution of annotations – and would therefore score perfectly on existing evaluation approaches – could make predictions that many users would disagree with when deployed.

The disagreement deconvolution is therefore important for evaluating a model's deployed performance regardless of how it was trained and regardless of the type of labels it produces – it asks for the single prediction that the model would ultimately make when deployed, and tells you how well that prediction would perform. Our goal is to precisely characterize the costs of using a single discrete decision rather than modeling the entire distribution.

2.3.2 De-noising and de-weighting. If existing work has shown that we can train models to accurately predict soft labels that represent annotators' distribution of opinions [53], why not simply use those predicted soft labels as indicators of how well each different label would perform, and compute metrics against them? The aim when predicting soft labels is essentially raw disaggregation – to reproduce the raw underlying annotator distribution. However, this raw distribution is contaminated by label noise and uncertainty. We argue that the distribution of primary labels, rather than raw labels, are the objects of interest.

We find these primary labels through a novel de-noising procedure. Existing de-noising procedures aim to remove noisy annotations by modeling their various sources [5, 19, 55]. Our de-noising procedure is unique in that it specifically aims to identify primary labels, or the label an annotator would provide most often were they to repeatedly annotate the same example over a period of time.

Finally, one common approach to the issue of annotator disagreement is to simply de-weight or remove disagreed upon examples from datasets, leaving only the items that feature strong agreement. Though this can improve test accuracy in hate speech and sentiment analysis [42, 69], we view this filtering technique as problematic because it removes the most challenging items and hides beliefs that the model's end-users still hold, while ignoring the fact that such examples will still exist when the model is deployed. The disagreement deconvolution demonstrates how models will perform over these otherwise hidden examples.

2.4 HCI and ML

Our work draws on a recent thread of research integrating human-centered methods into ML systems. Interactive machine learning seeks methods to integrate human insight into the creation of ML models (e.g., [3, 27]). One general thrust of such research is to aid the user in providing accurate and useful labels, so that the resulting model is performant [17]. Research has also sought to characterize best practices for designers and organizations developing such classifiers [2, 4].

End users struggle to understand and reason about the resulting classifiers. Many are unaware of their existence [26], and many others hold informal folk theories as to how they operate [22]. In response, HCI researchers have engaged in algorithmic audits to help hold algorithmic designers accountable and make their decisions legible to end users [59].

Our work extends this literature, focusing on ameliorating issues that developers and product teams face in reasoning about their models and performance [51]. To do so, we propose an alternative mental model for developers and product teams to reason about performance metrics: rather than evaluating against labels from aggregated pseudo-humans as is common today, our approach enables developers and product teams to think of performance metrics in terms of the proportion of the overall population that would agree with the classifier.

3 DISAGREEMENT DECONVOLUTION

Within each dataset of annotator labels, some disagreement is the result of labelers holding different opinions, meanwhile other disagreements are results of misunderstandings, errors, or even a momentary change of heart. Faced with both inter-annotator and intra-annotator disagreement, how should we determine whether a classifier made the right prediction? We take the position that classifiers should be judged against each annotator's *primary label*: the label they would provide the most often in an imagined scenario

where they label the same item attentively and repeatedly without memory of each prior label. In other words, even if an annotator might sometimes label an article as misinformation and sometimes not, we aim to identify the label they would apply most often if they completed the task in multiple simultaneous parallel worlds.

In this section, we describe the *disagreement deconvolution*, an algorithm designed to transform a dataset of annotations into one that allows sampling from the primary labels for each item, with random noise and non-primary labels removed. From such a dataset, we are able to compute disagreement-adjusted versions of any standard classification metric and the metric now treats each annotator's primary label as individual ground truth values. Under such a metric, there can be many ground truths per example; for example, "given each annotator's primary label, 75% of annotators would most likely believe this comment constitutes misinformation, and the other 25% would most likely believe that it does not constitute misinformation."

The disagreement decovolution is compatible with any standard classification metric, and for tasks with any number of classes. The input to the deconvolution is a dataset with multiple annotators labeling each item, commonly acquired from crowdsourcing services, which we refer to as the *raw disaggregated* dataset because it contains individual annotators' labels rather than a single aggregated (e.g., via majority vote) ground truth label. The output of the disagreement deconvolution is an estimated distribution of primary labels for each item in the dataset.

At a high level, our procedure:

(1) Estimates how often an annotator returns non-primary labels for a particular item, which we call p_{flip}
(2) Uses p_{flip} to generate an adjusted label distribution p^* of annotators' primary labels for that item
(3) Generates a new test set as a random draw from p^*
(4) Runs any performance metric over the new test set

For pedagogical clarity, we will begin by describing the high level ideas using the case where p_{flip} is given as an input to our algorithm, and defer the complexities of estimating p_{flip} to Section 4.

3.1 Estimate the population's distribution of primary labels from p_{flip}

We will begin by deriving how to transform an estimate of the probability that an annotator will return a label differing from the primary response for an item, p_{flip}, and the observed label distribution p, into a distribution if every annotator returned their primary label p^*. p is simply the distribution of responses for the item in the raw disaggregated dataset (e.g., four "harassment" labels and six "non-harassment" labels). We will defer the estimation of p_{flip} for now, and return to it once the overall procedure is clear.

The estimator that we propose subtracts p_{flip} from the observed label distribution (since a p_{flip}-fraction of the labels are expected to differ from the primary label), thresholds at zero, and re-normalizes the distribution such that the probability over all labels sums to one. Intuitively, we account for the fact that each label in the distribution has a p_{flip} chance of not being a primary label. We can view this procedure as interpolating between two regimes. On one hand, we have disagreggation when $p_{\text{flip}} = 0$ and our estimator returns the raw disaggregated distribution unmodified. On the other hand we have aggregation whenever p_{flip} is greater than the frequency of all non-primary labels. In this case our procedure removes all the probability mass from all labels but the primary one, and returns one consensus label for each example, equivalent to a majority vote aggregation. If p_{flip} is between these values, we can intuitively think of the procedure as "sharpening" the distribution by reducing the weight on the minority class, since a p_{flip} proportion of the minority responses were mistakes or other non-primary labels.

We can derive this estimator more formally under a particular generative model for label noise. Consider the general K-class classification problem with labels y and examples x. In this generative process, we first draw the primary label y^* from $p^*(y \mid x)$ and then draw an indicator z_{flip} that is positive with probability $p_{\text{flip}}(x)$ to determine whether we observe the primary label or some other random label y_{other} from the other $K-1$ choices.

Let Categorical(S, p) be the categorical distribution over elements of a set S with probability distribution p, then we can state the generative model as

$$\begin{aligned} y^* \mid x &\sim \text{Categorical}([K], p^*(y \mid x)) \\ y_{\text{other}} \mid y^* &\sim \text{Categorical}([K] \setminus \{y^*\}, \mathbf{1}/(K-1)) \\ z_{\text{flip}} \mid x &\sim \text{Bernoulli}(p_{\text{flip}}(x)) \\ y \mid y^*, y_{\text{other}}, z_{\text{flip}} &= y^*(1 - z_{\text{flip}}) + y_{\text{other}} z_{\text{flip}} \end{aligned}$$

Given the true $p_{\text{flip}} < 1/K$ and $p(y \mid x)$, we can identify $p^*(y \mid x)$ as

$$p^*(y \mid x) = \frac{p(y|x) - p_{\text{flip}}(x)/(K-1)}{1 - K p_{\text{flip}}(x)/(K-1)} \tag{1}$$

This follows directly from expanding the marginal distribution over y and solving for p^*,

$$p(y|x) = (1 - p_{\text{flip}}(x)) p^*(y \mid x) + (1 - p^*(y \mid x)) p_{\text{flip}}(x)/(K-1)$$

We can now directly plug in any estimate of p_{flip} to estimate $p^*(y \mid x)$. However, this estimator can be problematic, as estimation errors may result in $p_{\text{flip}}(x)/(K-1) > p(y \mid x)$ implying negative probabilities for $p^*(y \mid x)$ which is clearly impossible. We take the straightforward approach of constraining our noise estimate to $p_{\text{flip}}(x)/(K-1) < p(y \mid x)$, which results in the following estimator, Equation 2:

$$\hat{p}(y \mid x) := \frac{\max(p(y \mid x) - p_{\text{flip}}(x)/(K-1), 0)}{1 - K p_{\text{flip}}(x)/(K-1)}. \tag{2}$$

3.2 Sample a test set of primary labels

Once we have found our distribution of primary labels, we can sample our new test set. We can think of $p^*(y \mid x)$ as representing an estimated distribution of the primary labels for each example, with the random noise and non-primary annotations removed. From this distribution, we can randomly sample many annotations per example, and add each to our test set. We recommend drawing at least ten annotations per item, to obtain a reasonable approximation of the distribution of primary labels.

3.3 Run any metric to evaluate an oracle or existing model against the new test set

The newly sampled test set can now be used in any downstream evaluation as a noise-free test set consisting solely of primary labels. This means that we can compute disagreement-adjusted versions of any classifier evaluation metric (e.g. accuracy or ROC AUC) by computing the metric over the disaggregated labels in the sampled test set. Disaggregation treats each annotator's primary label as a potential ground truth label and ensures that models cannot achieve perfect performance simply by predicting the majority label for each example.

Examining the disaggregated dataset makes it clear that there are fundamental limits to the performance of *any* classifier. The best that a model can do on an example is predicting the majority label, but this will still incur errors over annotators whose primary labels differ from the majority. We show that it is possible to estimate an upper bound to any classifier's performance via an oracle classifier. The oracle classifier is one that is able to know the distribution of annotators' labels and always predicts the majority annotation for each example. We can construct this oracle classifier explicitly using any given dataset - we simply set the prediction of the oracle on each example to be the majority label.

The oracle is particularly useful, as it can be constructed with very few samples and used as a diagnostic when designing and evaluating data collection procedures. For instance, 1000 examples might be far too few examples to train a high-performance classifier but oracle classifier performance can be estimated reliably, since constructing an oracle classifier only requires that we be able to identify the majority label in each example. This enables researchers to collect small pilot datasets and easily identify inherent disagreement and the upper bound on potential classifier performance.

4 FINDING p_{flip}

Above, we assumed that we already knew p_{flip} and used it as input to the disagreement deconvolution. In this section, we discuss how to compute p_{flip}. We first discuss an ideal case in which we can directly compute p_{flip} for each individual annotator+example combination. As the data for such an ideal case would rarely be feasible to collect, we then discuss two methods that estimate p_{flip} using data many datasets would already contain.

4.1 Ideal computation

How can we estimate p_{flip}, the probability that a given annotation will flip away from the most common response the annotator would provide to the question? p_{flip} is a function of both the annotator and the example.

Our insight is that p_{flip} for a given annotator and example can be derived from an observed *test-retest disagreement rate*, the rate at which an annotator provides the same label when asked the same question twice. Intuitively, if an annotator disagrees with themselves in a binary task when shown the same example twice, then at least one of these annotations is not their primary label and can be considered either random noise or a non-primary label. On the other hand, if an annotator agrees with themselves, either both annotations represent their primary belief (occurring with probability $(1 - p_{\text{flip}})^2$, or both annotations are flipped from their primary belief (with probability p_{flip}^2). We call this test-retest self-disagreement rate $r_{\text{self-disagree}}$. So, if we observe a sufficiently large number of test-retests, we expect $r_{\text{self-disagree}}$ to follow

$$r_{\text{self-disagree}} = 1 - ((1 - p_{\text{flip}})^2 + p_{\text{flip}}^2)$$

To solve for p_{flip}, we simplify this into a quadratic equation

$$r_{\text{self-disagree}} = 1 - \left((1 - p_{\text{flip}})^2 + p_{\text{flip}}^2\right) = 2p_{\text{flip}} - 2p_{\text{flip}}^2$$

which we can then solve as Equation 3:

$$p_{\text{flip}} = \frac{1 - \sqrt{1 - 2r_{\text{self-disagree}}}}{2}. \tag{3}$$

The above approach is idealized and relies on every annotator providing several test-retests for each example. This is rarely feasible to collect, so we now introduce two methods to estimate p_{flip} using data that many machine learning datasets already contain. In the first estimate, we stratify test-retest annotations into buckets of similar examples. In the second estimate, we require no test-retest annotations at all, relying on singular value decomposition to determine whether an annotation is an annotator's primary label.

4.2 Stratified estimation

How can we estimate p_{flip}, without requiring many test-retest annotations for each example+annotator combination? We draw on the intuition that, across all annotators, some items are much easier to label than others. Some items, for example, may be clear harassment, and annotators are unlikely to flip on these; other items exist in a gray area, and annotators are much more likely to flip if they encounter the item again later. So, we base our estimate on the insight that if two examples are likely to have similar p_{flip} values, then we can combine their test-retest annotations to determine that p_{flip} value. We call this stratified estimation, or $p_{\text{flip\_strata}}$. In stratified estimation, we compute test-retest disagreement rates by aggregating test-retest annotations across examples that we expect to have, on average, similar test-retest disagreement rates.

Strata could potentially be defined by many variables that seem likely to predict $r_{\text{self-disagree}}$; how should we define them? We choose to define strata based on the overall level of disagreement for the item, which can be defined as the percentage of labels in the non-majority class. For example, if 60% of annotators label an item as harassment and 40% as non-harassment, the level of disagreement is .4. We choose this definition of strata under the assumption that the items that are the most disagreed upon will behave differently than those where (nearly) everyone agrees, which could potentially explain much of the variance in test-retest disagreement rates between examples.

The upside of this approach is that, unlike in the ideal computation, we can perform the disagreement deconvolution on examples or annotators for which we have few or no test-retest annotations, so long as there are other examples in the stratum that do have test-retest annotations.

This method introduces a new bias-variance tradeoff: defining strata bounds to be larger will result in a lower variance but higher bias, because the more examples it can include, the more stable

$r_{\text{self-disagree}}$ can be, but the less accurate it might be for a particular example. Small strata will have the reverse.

Once we have observed $r_{\text{self-disagree}}$, we can derive $p_{\text{flip\_}strata}$ using Equation 3. Then, for any example, we simply determine which strata in falls into, and the $p_{\text{flip\_}strata}$ we computed for that strata is the p_{flip} we use for this example. In practice, we find that this strata-based method of estimation is applicable to most datasets.

4.3 SVD estimation

While collecting test-retest data is a common practice in some fields for understanding dataset reliability, some datasets used for machine learning do not contain it (e.g., [6, 33]). How might we measure p_{flip} without test-retest data?

Consider the following thought experiment: what if we had access to an oracle that could predict each person's primary labels perfectly? In this case, *the error rate of this per-annotator predictor would exactly be p_{flip}*. Thus, we can view the problem of estimating p_{flip} as being equivalent to estimating the optimal per-annotator classifier.

Algorithms used for recommender systems, such as the singular value decomposition (SVD) of the item-annotation matrix can be thought of as per-annotator classifiers. We can therefore estimate p_{flip} using SVD, which we call $p_{\text{flip\_}svd}$. SVD relies on finding patterns across annotator behavior to predict an annotator's rating for each item. In this sense, it borrows signal from other similar annotators to fill in estimates of a annotator of interest. This approach seems particularly well-suited to many social computing problems because these tasks are often in domains where people often exhibit strong group effects such as political opinions [37], meaning many annotations will be somewhat stable and predictable between groups of people. On the other hand, SVD is poorly suited to predicting outliers: disagreement that is stable within an individual but cannot be attributed to groups.

Of course, beyond struggling with outliers, SVD's performance can vary depending on a variety of factors, and it will often be far from an oracle predictor of per-annotator primary labels. But the closer we can get to an oracle classifier, the more accurate our estimate of $p_{\text{flip\_}svd}$ will be. Therefore, we propose the following approach to using SVD to estimate the optimal per-annotator classification error.

Consider that the goal of the SVD predictor is to approximate the *optimal* per-annotator label prediction. In practice, that SVD models are far from optimal when evaluated on a held-out test set. One reason for this is that the existing datasets often do not contain enough annotations, with many annotators in the dataset rating very few examples (making it difficult to confidently assign annotators to latent groups) and many examples having few annotations. This results in substantially lower accuracy and higher $p_{\text{flip\_}svd}$ compared to test-retest estimates. So, as an alternative, we estimate $p_{\text{flip\_}svd}$ using the error rate of the SVD model on the *training* data. Using training (rather than test) error reverses the bias of the finite-sample effects, resulting in more optimistic $p_{\text{flip\_}svd}$ estimates that we found in our experiments closely match the test-retest based estimates. For instance, the mean $p_{\text{flip\_}svd}$ value across the entire Jigsaw toxicity dataset (discussed in the Measurements section) is .163 when computed using the training set, and .222 using a held-out test set. The more optimistic value is closer to the value we find using a stratified test-retest estimate, of .122. Estimating error via the training set is usually discouraged, as training set error is not an upper bound on generalization error and can lead to overfitting. However, these drawbacks are not inherently problematic in our setting as we do not care about upper bounding the generalization error of the SVD model, and only require that its performance provide accurate $p_{\text{flip\_}svd}$ estimates. For this approach to work well, it is critical to tune hyperparameters (such as the number of factors or epochs) on a held-out validation set because our approach relies on SVD generalizing to the task as best it possibly can. This tuning must be performed on a validation set because overfitting to the training set would result in accuracies that are far too high, meaning $p_{\text{flip\_}svd}$ estimates would be far too low.

Following this process, we recommend following the stratification estimation procedure to estimate more precise $p_{\text{flip\_}svd}$ values for specific items, where $p_{\text{flip\_}svd}$ is 1 - SVD accuracy for each stratum.

In summary, our proposed procedure is:

(1) Split the dataset into training and validation sets (no test set is needed).
(2) Create an SVD model on the training set, where users are annotators and examples are items.
(3) Tune hyperparameters using the validation set.
(4) Compute SVD's accuracy *over the training set.*
(5) Stratify examples and compute $p_{\text{flip\_}svd}$ as 1 - SVD accuracy for each stratum.

5 MEASUREMENTS

In this section, we apply the disagreement deconvolution to three social computing and two classic machine learning tasks. We sought out datasets for tasks that are broadly representative of popular social computing or classic ML tasks and common in discourse in social computing or ML literature, while remaining distinct from one another.

We use the disagreement deconvolution to compute disagreement-adjusted metrics for *oracle models*: models that always predict the majority aggregated annotation. Similar in concept to a Bayes optimal classifier [18], an oracle model's disagreement-adjusted scores are the highest possible scores that any classifier could ever hope to achieve on the task.

With a standard aggregated test set, an oracle model will always receive perfect scores on metrics such as accuracy, ROC AUC, precision and recall; whereas following the disagreement deconvolution, the oracle model will only get credit in proportion to the distribution of labelers who select the aggregated class as their primary class, which can be far from perfect. The oracle model's metrics can thus be understood as the percentage of the population that would assign credit (based on the metric) for making the most popular class prediction for each example. If even an oracle classifier cannot achieve high performance, then we urge caution when training and deploying a machine learning model for the task.

We evaluate these oracle models against test sets created with both the disagreement deconvolution and, as a baseline, raw disaggregation. For an oracle model, raw disaggregation establishes a

lower bound for disagreement-adjusted scores because it assumes that all annotations represent an annotator's primary label and does not remove errors. The fact that raw disaggregation is a lower bound follows from the observation that the disagreement deconvolution always sharpens the distribution toward the more popular class for each example.

We aim to answer the following questions in this analysis:

(1) What are the disagreement-adjusted scores of social computing tasks for theoretical oracle models?
(2) How do the estimates from the disagreement deconvolution compare to a raw disaggregated dataset ($p_{\text{flip}} = 0$)?
(3) Are there categorical differences in the amount of adjustment (disagreement) between social computing tasks and classic ML tasks?
(4) How closely does the SVD-based $p_{\text{flip\_}svd}$ approximate the more data intensive $p_{\text{flip\_}strata}$?

We'll first briefly describe our five datasets. To help illustrate the process of applying the disagreement deconvolution, we'll then walk through a worked example with the Jigsaw dataset. Finally, we'll report and discuss results from all five tasks.

5.1 Dataset Descriptions

5.1.1 Jigsaw toxicity. The Jigsaw Unintended Bias in Toxicity Classification dataset was developed by Google Jigsaw with the aim of supporting the development of machine learning classifiers to remove toxic comments from the web. The dataset consists of 1.8 million comments, each rated by 4–10 annotators whether or not the comment is toxic. Following common crowdsourcing practice [43], the dataset was collected by adaptively sampling more annotators for the examples with the highest uncertainty. This dataset contains 100,696 test-retest annotations, so we are able to estimate p_{flip} from stratified estimation.

Precision and recall are the most appropriate metrics for this task, given we are primarily interested in the positive class but the dataset is heavily imbalanced towards the negative class. We also include ROC AUC because it is the metric used in Jigsaw's popular Kaggle competition. The best published model on this task has a traditional ROC AUC of .95.

5.1.2 Credibility-Factors2020. This dataset aims to support the development of better tools for misinformation [9]. This dataset is a subset of a broader dataset of 2000 articles:[1] these 50 articles have been extensively annotated by crowd workers and climate experts for credibility on a 5-point Likert scale. Each example was annotated by 49 students, 26 Upwork workers, 3 science, and 3 journalism experts. We focus on the annotations created by the 26 Upwork workers. The dataset has a Fleiss Kappa score of 0.24.

This dataset does not include any test-retest annotations, so we used SVD to approximate p_{flip}. Though the dataset is very small in its number of examples, the fact that every example was labeled by all 26 Upworkers provides hope that SVD would be able to model most of the signal present in the data. Given the small size of this dataset, modeling all 5 discrete classes would be challenging, so we collapsed the 5-point Likert scale into a 3-point scale representing not credible, not sure, and credible. We use accuracy as our classification metric.

5.1.3 Get Another Label: PG13+. This dataset [61] is a classic dataset in crowdsourcing optimization, used to support machine learning models that detect adult content on webpages. The possible annotations are split into P, G, R, B, and X. The dataset[2] contains over 100k examples with between 1 and 30 annotations per example. We drop all examples with fewer than 3 labels. Like with the Jigsaw dataset, the number of annotators for each example grows with the uncertainty of the example: unanimous responses contain fewer labels. This dataset contains 5,840 test-retest annotations, so we are able to estimate p_{flip} from stratified estimation.

5.1.4 Word Sense Disambiguation. As a classic machine learning task in NLP, word sense disambiguation presents a sentence with a highlighted word and then requires an algorithm to choose which of several different meanings (senses) of the word are intended. This dataset[3] annotates a portion of the SemEval Word Sense Disambiguation Lexical Sample task.

The dataset had over 177 examples with 10 annotations per example. It did not contain test-retest annotations, so we used SVD to approximate p_{flip}.

5.1.5 CIFAR-10h. CIFAR-10 is a classic task in computer vision, focusing on image classification across 10 basic classes such as automobile, bird, or cat. The CIFAR-10h dataset [53] contains 10,000 CIFAR-10 images each labeled by 50 annotators. Given the large number of classes and balanced dataset, accuracy is a good metric for this task. This dataset does not contain test-retest annotations, so we computed $p_{\text{flip\_}svd}$.

5.2 Applying The Disagreement Deconvolution to Jigsaw

For illustrative purposes, we will walk through the application of the disagreement deconvolution to the Jigsaw dataset.

5.2.1 Estimate p_{flip}. The Jigsaw dataset contains 100,696 test-retest annotations, meaning we can potentially estimate p_{flip} using either the ideal computation or $p_{\text{flip\_}strata}$. Very few examples in this dataset have more than two test-retest annotations, indicating that $p_{\text{flip\_}strata}$ is more appropriate.

We now determine the width of our strata. Given the larger number of test-retest annotations, we choose buckets that increase monotonically by .05 from 0 to 1, creating 20 strata to capture many fine gradations between "high agreement" and "high disagreement"

We proceed as follows: for each strata, we compute the test-retest disagreement rate. For example, in the (.55, .6] strata, the test-retest disagreement rate $r_{\text{self-disagree}} = 0.291$. Applying this rate into Equation 3, which transforms $r_{\text{self-disagree}}$ into $p_{\text{flip\_}strata}$, we find $p_{\text{flip}} = 0.176$. We repeat this process for each strata, resulting in 20 $p_{\text{flip\_}strata}$ values.

5.2.2 Compute primary label distributions. Armed with our $p_{\text{flip\_}strata}$ values, we can now estimate the population's primary label distribution for each example. For instance, take an example with 10 total

[1] https://data.world/credibilitycoalition/2019-study

[2] https://github.com/ipeirotis/Get-Another-Label/tree/master/data

[3] https://sites.google.com/site/nlpannotations/

annotations, 6 annotators voting toxic and 4 voting non-toxic. In addition to $p_{\text{flip\_strata}}$ for the (.55, .6] strata, our formula also requires $p(y = 1 \mid x)$, which for this example is .6 because 60% of annotators provided this label. Plugging these values into Equation 2, we find .54. Similarly, if we repeat this process for the negative class, we find .28. We therefore estimate that for this example, 54% of annotations are for the toxic class, 28% for the non-toxic class, and 18% are non-primary labels. We repeat this process for every example to find the percent of primary labels that fall into each class.

5.2.3 Sample a new test set. Armed with our primary label distribution, we can now sample a new test set. We first need to choose a number of annotations to sample. As this number goes to infinity, we would exactly represent the distribution. To ensure the simulated test set remains a reasonable size, we select 10. For each example, we then use a weighted choice function to randomly select an annotation in {0, 1, non-primary label} with weights $\hat{p}(x|y = 0)$, $\hat{p}(x|y = 1)$, and $1 - (\hat{p}(x|y = 1) + \hat{p}(x|y = 0)$, and add annotations in classes 0 or 1 to the test set as a label for that example.

5.2.4 Apply existing metrics to the test set. Finally, armed with our new test set, we compute disagreement-adjusted performance results using any existing metric. While a typical test set only has one true value and one prediction per example, ours has ten: each of the true values was drawn from the stochastic objects for that example, while the predicted value remains the constant. We discuss our results later in this section.

5.2.5 Estimating p_{flip_svd}. The Jigsaw dataset contained test-retest data, meaning we could derive p_{flip} from observed test-retest disagreement. For illustrative purposes, we also report the SVD estimation method, $p_{\text{flip\_svd}}$. We use the Python library, Surprise, to train an SVD model, treating the example IDs as items, the annotator IDs as users, and the annotations as ratings. We split our dataset into a random 80/20 train/validation set because, as we note in the methods section, our approach is a rare instance in which the training set will itself function as our ultimate test set. We fine-tune two hyperparameters: number of epochs and number of factors using a basic grid search on the validation set. We then compute predictions over our training set, split each training example into strata (in this case, we use the same strata we defined above when we we estimated $p_{\text{flip\_strata}}$). Within each strata, we take all the predictions and compute their accuracy, resulting in one accuracy value per strata. Each strata's $p_{\text{flip\_svd}}$ is then $1 - accuracy$.

5.3 Results

Table 1 summarizes the results of computing traditional standard aggregated performance, raw disaggregated performance, and disagreement-adjusted performance for an oracle model for all five tasks. The standard aggregated performance is always a perfect score, so it serves to highlight the potentially drastic difference between a "perfect" model when we ignore disagreement and when we do not.

We first focus on the three social computing datasets: Jigsaw toxicity detection, credibility classification, and adult content rating. For the Jigsaw task, the disagreement-adjusted performance is dramatically lower than the perfect 1.0 for the standard evaluation regime, at .774 precision and .494 recall when we compute p_{flip} using stratified test-retest estimation. As a sanity check to establish the possible range we might expect from these scores, we can compute standard binomial confidence intervals over $r_{\text{self-disagree}}$ for each stratum and propagate the resulting 5th and 95th percentile $r_{\text{self-disagree}}$ through our algorithm, resulting in bounds of 0.773–0.781 precision and 0.477–0.496 recall.

The disagreement-adjusted performance is also dramatically lower than the perfect 1.0 for the standard evaluation regime for the credibility classification task, at 62.6% accuracy. Adult content rating comes in at a less severe but still low 79.5% accuracy. These are the highest possible scores that any classifier for these tasks could ever receive, when we take into account the primary labels of each annotator.

For comparison, we computed the same performance metrics using raw disaggregation. For the Jigsaw and credibility tasks, the disagreement deconvolution produced notably higher scores (.774 vs .710 precision for Jigsaw, and 62.6% vs 49.1% accuracy for credibility), indicating that some of the penalties from raw disaggregation represent error or non-primary labels. These results highlight the importance of using the disagreement deconvolution to obtain estimates that remove such annotations. In the adult content task, we saw that the difference was far smaller, at 79.1% vs 79.5%, indicating that nearly all of the disagreement between annotators is true disagreement in primary labels.

Our classic ML tasks, WSD and CIFAR-10h, were penalized far less by the disagreement deconvolution, indicating far less disagreement in the population about these tasks. Both of these tasks had high raw disaggregated performance to begin with, indicating that the large majority of annotators already agree with each example's aggregated label. WSD's disagreement-adjusted accuracy is 99.7%, compared to a 98.7% raw disaggregated accuracy, indicating very little disagreement in the dataset. In CIFAR-10h, disagreement-adjusted accuracy is 91.1% vs. 90.1% for raw disaggregation, suggesting more disagreement than WSD but very little error or inconsistency.

Comparing the SVD estimates of p_{flip} to the stratified estimates of p_{flip}, we confirm that SVD is a more optimistic estimate by observing that it leads to higher disagreement-adjusted scores. For example, our Jigsaw's disagreement-adjusted precision was .734 when we estimated p_{flip} using stratified estimate, and .806 using the SVD estimate.

Finally, to move beyond oracle models and understand how today's real models fare, we evaluated a Jigsaw model trained using the most upvoted architecture on the Jigsaw Kaggle competition's discussion page, an LSTM architecture created with Keras and gensim embeddings [38]. We find that this model achieves a ROC AUC of .973, precision of .527, and recall of .827 (precision and recall are computed with a soft label threshold of 0.5) over a standard aggregated test set. We note this ROC AUC is slightly higher than any competition submission scored, likely because the competition's scores were adjusted to balance overall performance with various aspects of unintended bias. When we evaluate this model against a test set created using the disagreement deconvolution, it achieves a ROC AUC of .726, precision of .514, and recall of .499, far lower performance than the metrics computed against an aggregated test set suggest.

Task	Test Set Created Using...	p_{flip} Estimator	Mean p_{flip}	ROC AUC	Precision	Recall	Accuracy
All tasks	Standard aggregation	N/A	0	1.0	1.0	1.0	1.0
Jigsaw	Disagreement deconvolution	Test-retest strata	0.122	0.734	0.774	0.494	N/A
Jigsaw	Disagreement deconvolution	SVD	0.163	0.806	0.806	0.623	N/A
Jigsaw	Raw disaggregation	N/A	0	0.697	0.710	0.413	N/A
Credibility	Disagreement deconvolution	SVD	0.302	N/A	N/A	N/A	62.6%
Credibility	Raw disaggregation	N/A	0	N/A	N/A	N/A	49.1%
PG13+	Disagreement deconvolution	Test-retest strata	0.054	N/A	N/A	N/A	79.5%
PG13+	Disagreement deconvolution	SVD	0.226	N/A	N/A	N/A	83.5%
PG13+	Raw disaggregation	N/A	0	N/A	N/A	N/A	79.1%
WSD	Disagreement deconvolution	SVD	0.100	N/A	N/A	N/A	99.7%
WSD	Raw disaggregation	N/A	0	N/A	N/A	N/A	98.7%
CIFAR-10h	Disagreement deconvolution	SVD	0.114	N/A	N/A	N/A	91.1%
CIFAR-10h	Raw disaggregation	N/A	0	N/A	N/A	N/A	90.1%

Table 1: Oracle models, for 3 social computing and 2 classic ML tasks, evaluated against test sets created from both disagreement deconvolution and raw disaggregation. While these models would all achieve perfect 1.0 scores against a standard aggregated test set, we find that our 3 social computing tasks perform far worse when we treat each individual annotator's primary label as ground truths. In two cases, they also perform notably better than naively computing metrics again raw disaggregation would suggest, indicating that annotators provided a substantial number of annotations that do not represent their primary labels. By comparison, we find that classic ML tasks suffer far less from the issue of disagreement, and that standard aggregated metrics are good estimates of their performance.

5.4 Validation

Finally, we aim to validate the disageement deconvolution's estimations. Validation requires that metrics computed over the disagreement deconvolution's transformation of a standard test set should be close to metrics computed over an idealized test set where we know, for each item, every annotator's primary label: the label each annotator would provide most often.

Obtaining such an idealized test set is typically infeasible; however, a subset of the Jigsaw dataset affords us an opportunity much closer to ideal than is typical. We subset Jigsaw to cases where annotators provided >= 3 annotations on the same example over a period of time, allowing us to treat the annotator's modal annotation for each item as their primary label to construct an idealized test set, $test_i$. A standard test set, $test_s$, randomly samples only one annotation per annotator for those same items. If the disagreement deconvolution (DD) works, then applying the DD to $test_s$ while holding out data from the idealized test set during DD's estimation process would find that Accuracy and ROCAUC results computed over a DD-transformed $test_s$ are closer to those metrics computed over $test_i$ than to $test_s$ without DD.

We indeed find that this is the case. The Jigsaw dataset contains 2662 instances (across $N = 577$ items) where at least one annotator provided >= 3 annotations, representing 2662 annotators' primary labels on those items. DD brings $test_s$ accuracy 27.4% closer to accuracy computed over $test_i$ (computed against our paper's oracle model's predictions), and ROCAUC 35.5% closer.

We can follow a similar protocol to examine the specific steps within the disagreement deconvolution: our sharpening procedure and p_{flip} estimation. For sharpening, we test whether the mean K-L divergence of $test_i$'s label distribution vs the disagreement deconvolution's sharpened $test_s$ distribution, is smaller than than the K-L divergence of $test_i$ vs the unsharpened $test_s$. We find this is true, with a K-L divergence of 0.38 compared to 0.66. For p_{flip}, we want the estimated p_{flip} to be reasonably close to the mean p_{flip} we observe from $test_i$, which is .101 (10% chance of flipping). We indeed find that $p_{\text{flip\_strata}}$ is 0.112, and $p_{\text{flip\_svd}}$ is .136, echoing our paper's claim that stratification is best when feasible.

6 DISCUSSION

In this section, we reflect on the limitations of our approach, as well as how designers and product teams might be impacted by it.

6.1 Limitations and Future Work

We believe that the disagreement deconvolution, by comparison to the standard evaluation approach procedure used today, presents a step forward in establishing useful performance estimates for social computing classifiers. As with any evaluation approach, there are several limitations and future directions worth discussing:

6.1.1 Weighting mistakes. The disagreement deconvolution maintains the interpretability characteristics of existing metrics, and enables us to adjust them for disagreements across annotators. In doing so, we effectively take the position that we would like to avoid

penalizing a model when an annotator cannot make up their mind about an example, which is why we give full credit if the model predicts an annotators' primary label. However, there are drawbacks to taking this position, which may render disagreement deconvolution a less realistic portrait of performance when deployed: even if an annotator would have accepted their non-primary label as correct, the model does not receive credit for predicting that non-primary label. For instance, when a fact checker disagrees with themselves when they see the same article again, the disagreement is probably not because they made a mistake or are randomly selecting labels, but because there is some probability mass they will answer with each label, depending on what parts of the article they're focusing on at the moment. In these cases, it is possible that an annotator might to some extent agree with either decision a classifier makes, though perhaps more strongly agreeing with one answer than another.

A future version of the disagreement deconvolution could address the challenge of evaluating models when annotators might accept multiple answers by addressing a related challenge: when a classifier is deployed, not all errors have the same cost. For instance, if a model predicts an annotator's non-primary label, it is possible that the annotator may still accept the non-primary label and the cost of that error might be quite low. Other errors, however, may have a very high cost. Future work could evaluate classifications using annotator-specific utility values for each decision a classifier makes [35], and weight errors accordingly.

6.1.2 Mathematical assumptions. Assumptions are a universal challenge inherent to every effort to create an evaluation procedure for ML tasks. Our aim with the disagreement deconvolution was to create an evaluation procedure that is more thoughtful about these assumptions for social computing tasks than are today's established evaluation approaches, yet remains feasible to compute with standard datasets. While not perfect, we believe that our evaluation procedure represents a significant step forward compared to the established procedures in use today.

We sought assumptions consistent with existing data collection efforts (e.g., a few test-retests on a subset of annotators) because we want this to be a widely usable measure. We can trade weaker assumptions for more extensive data collection. For instance, we assuming that labels other than the primary label would be chosen with the probability of $1/(K-1)$, and could remove this uniform $K-1$ assumption by asking for K times more test-retest data from each annotator. Similarly, we could avoid stratification by asking every annotator to perform many test-retests.

If test-retest rates are heterogeneous, sharpening may not reduce noise. However, *over*sharpening would require an unusual number of non-primary labels to independently arise on one item; this probability diminishes exponentially with the number of annotators. So, it is unlikely—and our validation above clarifies that sharpening improves the distribution's K-L divergence relative to an unsharpened baseline.

Finally, we note that an annotator's primary label may change over time, though most crowdsourcing tasks do not assume this happens within the limited timeframe of annotators' work.

6.1.3 Annotators representing end-users. A related limitation lies in a core assumption that often stymies ML systems when deployed: the annotators who create a dataset that is used to train and evaluate a model may not be representative of the stakeholders in that model when deployed. Though all evaluation approaches will suffer when this is not true, this issue can potentially be more pronounced when computing disagreement-adjusted metrics than when computing aggregated metrics, because aggregation can potentially mask some differences in the distributions.

6.1.4 Errors. Finally, while we provide guidelines as to which features of a dataset indicate disagreement deconvolution might produce accurate results for your dataset, and sanity-checks to help ensure reasonable results, we leave mathematical robustness claims that establish formal bounds of the errors disagreement deconvolution might make to future work. This includes potential errors introduced by our estimation approaches for p_{flip}.

6.2 Implications For Design

Our disagreement-adjusted scores demonstrate that even an oracle model would perform poorly on popular, state of the art toxicity and misinformation datasets, and that today's standard metrics dramatically overstate their performance.

Fundamentally, these classifiers are limited by the tasks they attempt to solve. Why have researchers been pursuing tasks that are doomed to such a fate? The fact that so many researchers and engineers are attempting to train these models using a standard supervised learning pipeline may be because the problem of annotator disagreement has been given inadequate attention given its impact. This may be due to a natural but misleading analogy that researchers and engineers make between these social computing tasks and classic ML tasks – tasks that supervised learning has been highly effective at, and for which we find the problem of disagreement is far less severe. The possible unconscious use of this analogy masks the impact of disagreement. Downstream, this masked disagreement results is overestimating the current capabilities of models and misattributing their limitations.

This is a natural analogy because, on the surface, these tasks seem similar: for both tasks, we require a large number of diverse examples, getting an individual decision from a human for each example is often fast, and they operate over text and images. But the analogy breaks down for social computing tasks because, even though getting a single individual annotator's decision is fast and easy, social computing classifiers are attempting to learn a model that makes decisions consistent with a community's values or norms, and many of these decisions would require an entire court case for each example — and even then, many would disagree with the court's ruling. This is why inter-rater agreement can be much lower for these social computing tasks.

We believe that using disagreement deconvolution is important because, if researchers and engineers are to overcome this misleading analogy and chart a path forward, we need clear metrics that can report the scale of the analogy's breakdown. Metrics that can compare the process of attempting to train traditional supervised learning models for these social computing tasks, with the process of training classic ML tasks. In this paper, we have enabled such metrics and demonstrated the results they can find.

6.3 Recommendations for usage and reporting

6.3.1 Recommended hyperparameters. The disagreement deconvolution requires choosing values for several hyperparameters. First is the choice of whether to compute p_{flip} using $p_{\text{flip\_}svd}$ or $p_{\text{flip\_}strata}$. Then, within both of these options, there are additional hyperparameters to choose from. In this section, we briefly describe recommended hyperparameters. These suggestions should only be used a starting point, and may vary significantly depending on the characteristic of the dataset. If the hyperparameters are not well-chosen, then it is possible to significantly inflate disagreement-adjusted scores. If the disagreement deconvolution is used in a scenario in which there may be temptation to artificially inflate scores, such as a competition, then hyperparameters for a particular dataset should be chosen ahead of time and held constant.

First, on the choice of $p_{\text{flip\_}svd}$ versus $p_{\text{flip\_}strata}$: as discussed in earlier sections, $p_{\text{flip\_}strata}$ is advised when the dataset contains sufficient data for stratification. Sufficient data means that we can compute reliable $r_{\text{self-disagree}}$ values for a large enough number of strata to find a reasonable approximation of $r_{\text{self-disagree}}$ for most examples. As a starting point: for typical datasets, we recommend at least ten strata and that each strata has at least 100 examples with at least 2 test-retests per example. Select the finest possible strata gradations for which at least that number of examples will be present in each strata. To investigate the impact of different levels of stratification, we recommend computing standard binomial confidence intervals over $r_{\text{self-disagree}}$ for each stratum and propagating both the resulting 5th and 95th percentile $r_{\text{self-disagree}}$ through our algorithm, which will establish bounds of how your disagreement-adjusted scores might change were you to re-collect the data in each strata. You can also inspect the difference between the 5th and 95th percentile $r_{\text{self-disagree}}$ of each strata; if the difference seems unreasonable for your task, then you should use strata with more data. Finally, we recommend repeating this process for multiple different numbers of strata to get a sense for the possible error bounds of using different strata values.

If there is not sufficient data available for $p_{\text{flip\_}strata}$, we instead suggest considering $p_{\text{flip\_}svd}$. Not all datasets, however, are appropriate for computing the SVD: there must be a reasonable number of instances where a few different annotators provided an annotation for the same example. A sanity check for whether the SVD is appropriate for your dataset is whether, on a held-out test set, a trained recommender system model using the SVD can achieve accuracy within the realm of what seems reasonable for your task. If the accuracy is far lower than you'd expect, this is a sign that the SVD is a poor fit for your dataset. When using the SVD, we recommend employing one of the popular libraries, which will often provide reasonable default hyperparameters.

6.3.2 Reporting. When a researcher's goal is to understand the extent to which users will agree with a model while deployed, then disagreement-adjusted scores exhibit the desired behavior, since users will disagree more with the results of one task (e.g., Jigsaw harassment) than another (Jigsaw hate speech). We recommend reporting these scores with "disagreement-adjusted" prepended to the metric's standard name; e.g. disagreement-adjusted precision.

However, if, as in today's ML metrics, a researcher's goal is to report a model's technical performance, then disagreement-adjusted metrics introduce a new challenge: it is difficult to compare them across multiple different tasks/datasets, because the best possible performance of each task is inherently limited by the disagreement present in the dataset. If a researcher would like to make disagreement-adjusted scores comparable between tasks, we recommend normalizing disagreement-adjusted scores against oracle level performance (their theoretical maximum). The difference between these normalized technical performance metrics and traditional technical performance metrics is that the normalized scores will 1) use our de-noising procedure to focus on primary annotations, and 2) place more weight on the test set examples that have more agreement, whereas traditional metrics give each example equal weight (which, as we discuss earlier in this paper, is not a valid approach to evaluating the user experience). In cases where both user experience and technical performance is of interest, we recommend reporting both of the above numbers.

6.4 Additional ethical considerations

Our approach introduces its own ethical tradeoffs. As described currently, the deconvolution assigns each annotator equal weight in the metric, a one-person-one-vote strategy that will disadvantage minority populations [63]. A participatory or value-centered approach would instead invite stakeholders to the table to help articulate how the community should weigh each annotator's beliefs. We will work to mitigate these issues by introducing techniques to explicitly up-weigh viewpoints from under-represented groups.

In addition, introducing any metric raises the question, "What second-order consequences will arise from an organization blindly hillclimbing on this metric?" One upside we hope to achieve is that organizations and developers will recognize when a classifier is doomed to failure, because even an oracle classifier has unacceptable performance. However, the most clear risk is again the tyranny of the majority. The most efficient way to increase a disagreement-adjusted metric is for the classifier to predict correctly for members of the majority group, because they are most numerous in the dataset. This happens today already, since the majority group dominates the aggregated labels. While our approach does not solve this issue, it does have one significant upside compared to aggregated labels: modelers are forced to confront the extent to which the minority beliefs fundamentally limit their model's performance, a fact that standard aggregated metrics hide.

7 CONCLUSION

Social computing tasks, and likely many others in human-computer interaction as well, are not clean perceptual-level tasks where nearly every annotator will agree on every label. Instead, social computing tasks often attempt to categorize the messy realities of our lives [12]. With that messiness comes contestation, disagreement, and deliberation. The disagreement deconvolution is an attempt to bridge the realities of machine learning, which requires a meaningful and interpretable evaluation metric, with the realities of social computing tasks, which require acknowledgment of the inherent disagreement in the task. We observe that this approach drastically reduces reported performance on several tasks. If successful, we hope that this approach will help developers make more informed decisions about training and deploying classifiers in these contexts.

ACKNOWLEDGEMENTS

We thank Jane L. E, Harmanpreet Kaur, Ranjay Krishna, Leon A. Gatys, Amy X. Zhang, Xinlan Emily Hu, and James Landay for insightful discussions and support. We thank the reviewers for their helpful comments and suggestions. Mitchell L. Gordon was supported by the Apple Scholars in AI/ML PhD fellowship and a Junglee Corporation Stanford Graduate Fellowship. Kaitlyn Zhou was supported by a Junglee Corporation Stanford Graduate Fellowship. This research is supported by the Hasso Plattner Institute Design Thinking Research Program. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

REFERENCES

[1] Ali Alkhatib and Michael Bernstein. 2019. Street-level algorithms: A theory at the gaps between policy and decisions. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300.

[3] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *Ai Magazine* 35, 4 (2014), 105–120.

[4] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-AI interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–13.

[5] Alexandry Augustin, Matteo Venanzi, J Hare, A Rogers, and NR Jennings. 2017. Bayesian aggregation of categorical distributions with applications in crowdsourcing. AAAI Press/International Joint Conferences on Artificial Intelligence.

[6] Jutta Backhaus, Klaus Junghanns, Andreas Broocks, Dieter Riemann, and Fritz Hohagen. 2002. Test–retest reliability and validity of the Pittsburgh Sleep Quality Index in primary insomnia. *Journal of psychosomatic research* 53, 3 (2002), 737–740.

[7] Barbara A Bailar. 1968. Recent research in reinterview procedures. *J. Amer. Statist. Assoc.* 63, 321 (1968), 41–63.

[8] Paul M Barrett. 2020. Who Moderates the Social Media Giants? *Center for Business* (2020).

[9] Md Momen Bhuiyan, Amy X. Zhang, Connie Moon Sehat, and Tanushree Mitra. 2020. Investigating Differences in Crowdsourced News Credibility Assessment: Raters, Tasks, and Expert Criteria. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW2, Article 93 (Oct. 2020), 26 pages.

[10] Reuben Binns, Michael Veale, Max Van Kleek, and Nigel Shadbolt. 2017. Like trainer, like bot? Inheritance of bias in algorithmic content moderation. In *International conference on social informatics*. Springer, 405–415.

[11] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*. 491–500.

[12] Geoffery C. Bowker and Susan Leigh Star. 2000. *Sorting Things out: Classification and Its Consequences.* MIT Press, Cambridge, MA, USA.

[13] Jonathan Bragg, Mausam, and Daniel S. Weld. 2018. Sprout: Crowd-Powered Task Design for Crowdsourcing. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 165–176.

[14] Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 conference on empirical methods in natural language processing*. 286–295.

[15] Robyn Caplan and Tarleton Gillespie. 2020. Tiered governance and demonetization: The shifting terms of labor and compensation in the platform economy. *Social Media+ Society* 6, 2 (2020), 2056305120936636.

[16] Stevie Chancellor, Jessica Annette Pater, Trustin Clear, Eric Gilbert, and Munmun De Choudhury. 2016. # thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 1201–1213.

[17] Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2334–2346.

[18] Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J Dembczynski. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 279–286.

[19] Zhendong Chu, Jing Ma, and Hongning Wang. 2020. Learning from Crowds by Modeling Common Confusions. arXiv:cs.LG/2012.13052

[20] John Joon Young Chung, Jean Y Song, Sindhu Kutty, Sungsoo Hong, Juho Kim, and Walter S Lasecki. 2019. Efficient elicitation approaches to estimate collective crowd answers. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–25.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[22] Michael A DeVito, Jeremy Birnholtz, Jeffery T Hancock, Megan French, and Sunny Liu. 2018. How people form folk theories of social media feeds and what it means for how we study self-presentation. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–12.

[23] Anca Dumitrache. 2015. Crowdsourcing disagreement for collecting semantic annotation. In *European Semantic Web Conference*. Springer, 701–710.

[24] Anca Dumitrache, Lora Aroyo, and Chris Welty. 2017. Crowdsourcing ground truth for medical relation extraction. *arXiv preprint arXiv:1701.02185* (2017).

[25] Anca Dumitrache, Lora Aroyo, and Chris Welty. 2018. Capturing ambiguity in crowdsourcing frame disambiguation. *arXiv preprint arXiv:1805.00270* (2018).

[26] Motahhare Eslami, Aimee Rickman, Kristen Vaccaro, Amirhossein Aleyasen, Andy Vuong, Karrie Karahalios, Kevin Hamilton, and Christian Sandvig. 2015. "I always assumed that I wasn't really that close to [her]" Reasoning about Invisible Algorithms in News Feeds. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 153–162.

[27] Jerry Alan Fails and Dan R Olsen Jr. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*. 39–45.

[28] Gösta Forsman and Irwin Schreiner. 2004. The design and analysis of reinterview: an overview. *Measurement errors in surveys* (2004), 279–301.

[29] Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. 2020. Evaluating nlp models via contrast sets. *arXiv preprint arXiv:2004.02709* (2020).

[30] Timnit Gebru. 2020. Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3609.

[31] Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. 2018. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*. 1–6.

[32] Tarleton Gillespie. 2018. *Custodians of the Internet: Platforms, content moderation, and the hidden decisions that shape social media.* Yale University Press.

[33] Louis Guttman. 1945. A basis for analyzing test-retest reliability. *Psychometrika* 10, 4 (1945), 255–282.

[34] Louis Guttman. 1946. The test-retest reliability of qualitative data. *Psychometrika* 11, 2 (1946), 81–95.

[35] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 159–166.

[36] Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. 2009. Data quality from crowdsourcing: a study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing*. 27–35.

[37] Maurice Jakesch, Moran Koren, Anna Evtushenko, and Mor Naaman. 2019. The Role of Source and Expressive Responding in Political News Evaluation. (2019).

[38] Jogsaw. [n.d.]. Jigsaw Unintended Bias in Toxicity Classification. https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/overview

[39] Charles D Jones. [n.d.]. *Accuracy of 1970 Census Population and Housing Characteristics as Measured by Reinterviews.*

[40] Jeremy Kahn. 2020. Can Facebook's new A.I. banish Pepe the Frog? https://fortune.com/2020/05/12/facebook-a-i-hate-speech-covid-19-misinformation/

[41] Sanjay Kairam and Jeffrey Heer. 2016. Parting crowds: Characterizing divergent interpretations in crowdsourced annotation tasks. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. 1637–1648.

[42] Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, et al. 2018. Sentiment analysis: It's complicated!. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 1886–1895.

[43] John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald. 2010. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, Vol. 2126. 22–32.

[44] Tong Liu, Akash Venkatachalam, Pratik Sanjay Bongale, and Christopher Homan. 2019. Learning to predict population-level label distributions. In *Companion Proceedings of The 2019 World Wide Web Conference*. 1111–1120.

[45] Elizabeth Lucas, Cecilia O Alm, and Reynold Bailey. 2019. Understanding Human and Predictive Moderation of Online Science Discourse. In *2019 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*. IEEE, 1–5.

[46] Kaitlin Mahar, Amy X. Zhang, and David Karger. 2018. Squadbox: A Tool to Combat Email Harassment Using Friendsourced Moderation. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13.

[47] VK Chaithanya Manam and Alexander J Quinn. 2018. Wingit: Efficient refinement of unclear task instructions. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*.

[48] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635* (2019).

[49] Josh Andres Zahra Ashktorab Narendra Nath Joshi Michael Desmond Aabhas Sharma Kristina Brimijoin Qian Pan Evelyn Duesterwald Casey Dugan Michael Muller, Christine Wolf. 2021. Designing Ground Truth and the Social Life of Labels. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2021).

[50] Tanushree Mitra and Eric Gilbert. 2015. Credbank: A large-scale social media corpus with associated credibility annotations.

[51] Kayur Patel, James Fogarty, James A Landay, and Beverly L Harrison. 2008. Examining Difficulties Software Developers Encounter in the Adoption of Statistical Machine Learning.. In *AAAI*. 1563–1566.

[52] Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics* 7 (2019), 677–694.

[53] Joshua C Peterson, Ruairidh M Battleday, Thomas L Griffiths, and Olga Russakovsky. 2019. Human uncertainty makes classification more robust. In *Proceedings of the IEEE International Conference on Computer Vision*. 9617–9626.

[54] Danny Pfeffermann and Calyampudi Radhakrishna Rao. 2009. *Sample surveys: design, methods and applications*. Elsevier.

[55] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[56] Sarah T. Roberts. 2020. Fewer Humans Are Moderating Facebook Content. That's Worrying. https://slate.com/technology/2020/04/coronavirus-facebook-content-moderation-automated.html

[57] Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118* (2017).

[58] Joni Salminen, Fabio Veronesi, Hind Almerekhi, Soon-Gvo Jung, and Bernard J Jansen. 2018. Online Hate Interpretation Varies by Country, But More by Individual: A Statistical Analysis Using Crowdsourced Ratings. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 88–94.

[59] Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort. 2014. Auditing algorithms: Research methods for detecting discrimination on internet platforms. *Data and discrimination: converting critical concerns into productive inquiry* 22 (2014).

[60] Mike Schaekermann, Joslin Goh, Kate Larson, and Edith Law. 2018. Resolvable vs. irresolvable disagreement: A study on worker deliberation in crowd work. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19.

[61] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. 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*. 614–622.

[62] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter* 19, 1 (2017), 22–36.

[63] Harini Suresh and John V Guttag. 2019. A framework for understanding unintended consequences of machine learning. *arXiv preprint arXiv:1901.10002* (2019).

[64] Keith Trnka, John McCaw, Debra Yarrington, Kathleen F McCoy, and Christopher Pennington. 2009. User interaction with word prediction: The effects of prediction quality. *ACM Transactions on Accessible Computing (TACCESS)* 1, 3 (2009), 1–34.

[65] Joseph D Tucker, Suzanne Day, Weiming Tang, and Barry Bayus. 2019. Crowdsourcing in medical research: concepts and applications. *PeerJ* 7 (2019), e6762.

[66] Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572* (2018).

[67] Alex Hai Wang. 2010. Detecting spam bots in online social networking sites: a machine learning approach. In *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 335–342.

[68] Jing Wang and Xin Geng. [n.d.]. Classification with Label Distribution Learning.

[69] Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*. 138–142.

[70] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 159–168.

[71] Biqiao Zhang, Georg Essl, and Emily Mower Provost. 2017. Predicting the distribution of emotion perception: capturing inter-rater variability. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. 51–59.

[72] Xinyi Zhou and Reza Zafarani. 2018. Fake news: A survey of research, detection methods, and opportunities. *arXiv preprint arXiv:1812.00315* (2018).