Managing Crowdsourcing Workers

Jing Wang
jwang5@stern.nyu.edu

Panagiotis G. Ipeirotis
panos@stern.nyu.edu

Foster Provost
fprovost@stern.nyu.edu

Department of Information, Operations, and Management Sciences
Leonard N. Stern School of Business, New York University

ABSTRACT

The emergence of online crowdsourcing services such as Amazon Mechanical Turk, presents us huge opportunities to distribute micro-tasks at an unprecedented rate and scale. Unfortunately, the high verification cost and the unstable employment relationship give rise to opportunistic behaviors of workers, which in turn exposes the requesters to quality risks. Currently, most requesters rely on redundancy to identify the correct answers. However, existing techniques cannot separate the true (unrecoverable) error rates from the (recoverable) biases that some workers exhibit, which would lead to incorrect assessment of worker quality. Furthermore, massive redundancy is expensive, increasing significantly the cost of crowdsourced solutions.

In this paper, we present an algorithm that can easily separate the true error rates from the biases. Also, we describe how to seamlessly integrate the existence of "gold" data for learning the quality of workers. Next, we bring up an approach for actively testing worker quality in order to quicky identify spammers or malicious workers. Finally, we present experimental results to demonstrate the performance of our proposed algorithm.

1. INTRODUCTION

Crowdsourcing has emerged over the last few years as an important new labor pool for a variety of tasks[4], ranging from micro-tasks on Mechanical Turk to big innovation contests conducted by Netflix and Innocentive. Amazon Mechanical Turk (AMT) today dominates the market for crowdsourcing micro-tasks that are trivial to humans, but challenging to computer programs. The requesters can post tasks, such as image tagging, language translation, event annotation, and workers complete them and get compensated in the form of micro-payments. The tremendous potential for labor supply makes it possible to get tasks done very quickly.

Despite the promise, significant challenges remain. Since it is unlikely that requesters can verify the quality of every submitted answer manually and the employer-employee relationship between requesters and workers are usually one-time, workers have incentives to give crappy answers with almost no effort. The presence of spammers or malicious workers undoubtedly harms the scalability and robustness of online markets, which makes effective spammer detection mechanisms very valuable.

One commonly known approach for dealing with this problem is to use "gold" data. Typically, employers insert a small percentage of questions for which they already know the answers into their tasks, and see how the performance of workers. Another solution is to rely on repeated labeling: simply ask multiple workers to complete the same task and use majority voting to identify the correct answers. Unfortunately, since repeated labeling is rather costly, this would negate the advantage of crowdsourcing.

A better use of redundancy is to identify the correct answers of each task and measure the labeling quality of each worker at the same time. Dawid and Skene [2] first proposed a solution based on an expectation maximization algorithm which would give us a "most-likely" answer for each task and a "confusion matrix" for each worker, listing the error probabilities for the worker. From the confusion matrix we can directly measure the overall error rate for each worker as the sum of the non-diagonal elements of the confusion matrix (properly weighted by the priors): this results in a single, scalar value as the quality score for each worker. However, this approach would underestimate the quality of workers who give consistently and predictably incorrect answers. The answers given by inherently biased workers are valuable since we can extract useful information from their seemingly low-quality labels. On the other hand, for skewed datasets, spammers can achieve low error rates by classifying all the examples into the majority class.

This paper makes several contributions to the learning of worker quality. First, we present an algorithm to separate the unrecoverable error rate from recoverable bias. The output of our algorithm is a scalar score representing the inherent quality of each worker. Second, we describe how to integrate the usage of "gold" data for learning the quality of workers and discuss its virtues and defects. Third, we bring up an *active testing* approach to decide when and how to test workers using an expected utility framework. Finally, we show on synthetic data that our strategy outperforms the commonly used, non-active strategies.

2. RELATED WORK

The widespread of spammers makes it necessary to design some mechanism which can accurately estimate the worker quality and reject or block low-quality workers accordingly.

To measure the quality of the workers, we can insert some tasks with "gold" labels into the stream of assigned HITs, and then compute the error rate of each worker to detect the spammers. When the true label is not available or very expensive to acquire, a common approach is to elicit multiple labels from imperfect labelers. Sheng et al. [6] talked about how to do repeated labeling in a way which accounts for both label uncertainty and model uncertainty. But they did not estimate the quality of workers.

Dawid and Skene [2] presented an expectation maximization algorithm to estimate the error rates of observers when all observers see all available patients. Bayesian versions of the algorithm were recently proposed by Raykar et al. [5] and by Carpenter [1]. The algorithm iterates until convergence, following two steps: (1) estimates the true response for each patient, using records given by all the observers, accounting for the error-rates of each observer; and (2) estimates the error-rates of observers by comparing the submitted records to estimated true response. The final output of the Dawid&Skene algorithm is estimated true response for each patient and estimated error-rate represented by "confusion matrix" for each observer.

Whitehill et al. [9] presented a probabilistic model to simultaneously infer the label of each image, the expertise of each labeler, and the difficulty of each image. Their assumption is that the log odds for the obtained labels being correct are a bilinear function of the difficulty of the label and the expertise of the labeler. Welinder et al. [8] proposed a generative Bayesian model in which each annnotator is a multidimensional entity with variables representing competence, expertise and bias. They also described an inference algorithm to estimate the properties of the data being labeled and the annotators labeling them.

The common objective across all the approaches above is that they only estimate error rate for each worker. Unfortunately, as we will see, the error-rate alone cannot sufficiently measure the underlying value of a worker.

3. WORKER QUALITY ESTIMATION: THE BASICS

3.1 Notations

The task in our case is a multiple choice question. There are N objects, $o_1, \ldots, o_N$, each being associated with a *latent* true class label $T(o_n)$, picked from one of the L different labels. Each object is annotated by one or more of the K workers, each having a varying degree of quality. To measure the quality of each worker, the algorithm endows each worker (k) with a *latent* "confusion matrix" $\pi^{(k)}_{ij}$, which gives the probability that worker (k), when presented with an object in class i, will classify the object into class j.

3.2 EM Algorithm

Algorithm 1 presents a sketch of the process. The algorithm iterates between estimating the correct labels $T(o_n)$ for each of the objects, and estimating the error rates $\pi^{(k)}_{ij}$ for each worker. Note here, we relax the assumption that all the objects are labeled by all the available workers. So in each iteration, we only consider relevant variables when updating the estimates.

3.3 Limitations

The algorithm works well when we are only interested in the error rate $\pi^{(k)}_{ij}$ for each worker. However, it provides insufficient information when our objective is to reject or block workers accordingly. A naive method is to simply sum up the non-diagonal entries of the matrix $\pi^{(k)}$, weighting each error rate by the estimated prior of each class. Unfortunately, this approach would wrongly reject biased but careful workers. For example, consider the following example:

EXAMPLE 1. *Consider two workers that label web sites into two classes:* porn *and* notporn. *Worker A is* always *incorrect: labels all* porn *web sites as* notporn *and vice versa. Worker B classifies all web sites, irrespectively of their true class, as* porn. *Which of the two workers is better? A simple error analysis indicates that the error rate of worker A is 100%, while the error rate of worker B is "only" 50%.*[1] *However, it is easy to see that the errors of worker A are easily reversible, while the errors of worker B are irreversible. In fact, worker A is a perfect worker, while worker B is a spammer.* □

```
  Input: Labels $l[k][n]$ from worker $(k)$ to object $o_n$,
  Output: Confusion matrix $\pi^{(k)}_{ij}$ for each worker $(k)$, Correct
          labels $T(o_n)$ for each object $o_n$, Class priors $Pr\{C\}$
          for each class $C$
1 Initialize error rates $\pi^{(k)}_{ij}$ for each worker $(k)$ (e.g., assume each
  worker is perfect);
2 Initialize correct label for each object $T(o_n)$ (e.g., using
  majority vote);
3 while not converged do
4     Estimate the correct label $T(o_n)$ for each object, using the
      labels $l[\cdot][n]$ assigned to $o_n$ by workers, weighting the votes
      using the error rates $\pi^{(k)}_{ij}$;
5     Estimate the error rates $\pi^{(k)}_{ij}$, for each worker $(k)$, using
      the correct labels $T(o_n)$ and the assigned labels $l[k][n]$;
6     Estimate the class priors $Pr\{C\}$, for each class $C$;
7 end
8 return Estimated error rates $\pi^{(k)}_{ij}$, Estimated correct labels
  $T(o_n)$, Estimated class priors $Pr\{C\}$
```

Algorithm 1: The EM algorithm for worker quality estimation.

So, naturally a question arises: How can we separate low-quality workers from high-quality, but biased, workers? On the other hand, strategic spammers can easily avoid detection by classifying almost all the examples into the majority class, especially when the dataset is highly imbalanced.

EXAMPLE 2. *Consider the same problem as before, but with a skewed class distribution: 95% of the web sites fall in the category of* notporn, *while only 5% of the web sites are* porn. *A strategic worker C classifies all web sites, irrespectively of their true class, as* notporn. *By doing so, he can achieve a low error rate of 5%, and will be considered as a high quality worker if we employ the simple error analysis.* □

So another question arises: How can we identify the strategic spammers? We examine these questions next.

4. SEPARATING ERROR AND BIAS

After running the EM algorithm, we have some reasonably accurate estimates of the error rates $\pi^{(k)}_{ij}$ for each worker. How can we estimate from these values the intrinsic, non-recoverable error rate?

We start with the following observation: Each worker assigns a "hard" label to each object. Using the error rate for this worker, we can transform this assigned label into a "soft" label, which is the best possible estimate that we have for the *true* label assignment. So, if we have L possible classes and the worker assigns class j as a label to an object, we can transform this "hard" assigned label into the "soft" label:

$$\left\langle \pi^{(k)}_{1j} \cdot Pr\{C=1\}, \ldots, \pi^{(k)}_{Lj} \cdot Pr\{C=L\} \right\rangle \qquad (1)$$

where $\pi^{(k)}_{ij}$ is the probability that worker (k) classifies into class j, an object that in reality belongs to class i, $Pr\{C=i\}$ is the prior that the object will belong to class i. We should note that the quantities above need to be normalized by dividing

[1] Assume, for simplicity, equal priors for the two classes.

them with

$$Pr\{AC = j\} = \sum_{i}^{L} \pi_{ij}^{(k)} \cdot Pr\{C = i\} \tag{2}$$

the probability that worker (k) assigns label j to any object.

EXAMPLE 3. *Take the case of worker A from the previous example. When this worker assigns a label of* Porn *(assume that porn is class 1), then the corresponding soft label has all the "probability mass" in the* NotPorn *category:*

$$\underbrace{\begin{pmatrix} 1 \\ 0 \end{pmatrix}}_{\textit{Assigned: Porn}} \Rightarrow \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\textit{Corrected to: NotPorn}}$$

On the contrary, for worker B, who always assigns porn*, the corresponding corrected soft label does not give us any information; the soft label simply says that the best guess are simply the class priors:*

$$\underbrace{\begin{pmatrix} 1 \\ 0 \end{pmatrix}}_{\textit{Assigned: Porn}} \Rightarrow \underbrace{\begin{pmatrix} Pr\{C = 1\} \\ Pr\{C = 2\} \end{pmatrix}}_{\textit{Corrected to: Class priors}}$$

□

So, what can we do with these soft labels? The basic idea is to estimate the *expected cost of a soft label*. To estimate the cost of a soft label, we need to consider the misclassification costs. In the simplest case, we have a cost of 1 when an object is misclassified, and 0 otherwise. In a more general case, we have a cost c_{ij} when an object of class i is classified into category j.

LEMMA 1. *Given the classification costs c_{ij} and a soft label $\mathbf{p} = \langle p_1, p_2, \ldots, p_L \rangle$, the expected cost of the soft label $\mathbf{p}$ is:*

$$ExpCost\,(\mathbf{p}) = \sum_{i=1}^{L} \sum_{j=1}^{L} p_i \cdot p_j \cdot c_{ij} \tag{3}$$

□

The proof is rather simple. The expected classification cost is equal to the probability of classifying the object in class i (which is p_i), multiplied by the probability of the object belonging to class j in reality (which is p_j), multiplied with the associated cost of classifying an object of class j into class i (which is c_{ji}). Summing across all classes, we have the result above.

The expected cost of a soft label reveals how much information we can get from a particular assigned label by a worker. Therefore, it is valuable in helping us assess the usefulness of workers. However, we should notice that this is not the minimum cost we would have, given the knowledge of the soft label. We call the cost incurred by always assigning the example to the class which yields the minimum cost the *minimized cost of a soft label*. The minimized cost can help us make the best decision of classification in single labeling. However, as explained later, it does not have some nice properties of expected cost.

LEMMA 2. *Given the classification costs c_{ij} and a soft label $\mathbf{p} = \langle p_1, p_2, \ldots, p_L \rangle$, the expected cost of the soft label $\mathbf{p}$ is:*

$$MinCost\,(\mathbf{p}) = \min_{1 \le i \le L} \sum_{j=1}^{L} p_j \cdot c_{ij} \tag{4}$$

□

```
    Input: Error rates $\pi_{ij}^{(k)}$ for each worker, Misclassification costs
           $c[i][j]$, Class priors $Pr\{C\}$
    Output: Expected for each worker
 1  foreach Worker $(k)$ do
 2      Estimate how often the worker $(k)$ assigns label $l$
        ($Pr\{AC = l\}$), using Eq. 2;
 3      $Cost[k] = 0$;
 4      foreach Label $l$, assigned with probability $Pr\{AC = l\}$ do
 5          Using Eq. 1, compute the soft label $\mathbf{soft}^{(k)}(l)$ that
            corresponds to label $l$ assigned by worker $(k)$;
 6          Using Eq. 3, compute $Cost(\mathbf{soft}^{(k)}(l))$ for the soft
            label;
 7          $Cost[k]$ += $Cost(\mathbf{soft}^{(k)}(l)) \cdot Pr\{AC = l\}$;
 8      end
 9  end
10  return $Cost[k]$ for each worker $(k)$
```

Algorithm 2: Estimating the Expected Cost of each Worker

The results illustrate that workers with error rate matrices that generate "soft" labels with probability mass concentrated into a single class (i.e., certain "posterior" labels) will tend to have low estimated cost, as the product of Equation 3 will be close to 0. On the contrary, workers that tend to generate "soft" labels that are spread out across classes (i.e., uncertain "posterior" labels) will tend to have high associated costs.

EXAMPLE 4. *Consider the costs for the workers A and B from the previous examples. Assuming equal priors across classes, and $c_{ij} = 1$, if $i \neq j$ and $c_{ij} = 0$, if $i = j$, we have the following: The cost of worker A is 0, as the soft labels generated by A are $\langle 0, 1 \rangle$ and $\langle 1, 0 \rangle$. For worker B, the cost is 0.5 (the maximum possible) as the soft labels generated by B are all $\langle 0.5, 0.5 \rangle$ (i.e., highly uncertain).* □

Given that we know how to compute the expected cost for each label, we can now easily estimate the expected cost for each worker. We first compute the priors $Pr\{AC = i\}$ (see Equation 2), which is the prior probability of the worker assigning label i to an object. Then we compute the "soft label" that corresponds to the assigned label (see Equation 1). Given the soft label, we use Equation 3 to compute its expected cost. Now, knowing how often the worker assigns a label and the expected cost, we can compute the average expected cost of each worker. Algorithm 2 illustrates the process.

As expected, perfect workers will have a cost of zero and random workers or spammers will have high expected costs. Notice, as illustrated in the example above, that it is not necessary for a worker to return the correct answers in order to have low costs! As long as the errors are predictable and reversible, the worker is assigned a low expected cost.

This tends to resolve quite a few issues with online workers that exhibit systematic biases in their answers but also put a lot of effort in coming up with the answers. Prior approaches that relied on agreement generate a significant number of rejections for such workers, which in turn alienates such high-quality workers, and discourages them from working with employers that rely on worker agreement. The proposed algorithm alleviates these concerns.

We define a *spammer* as a worker who always assigns labels randomly, regardless of what the true class is. The quality score of a worker is computed by comparing his expected cost with the cost of a spammer: $QualityScore_i = 1 - \frac{ExpCost_i}{ExpCost_{spammer}}$. We can also compute the quality score based on minimized cost in the same way. Our results indicate that the two give pretty similar results.

5. GOLD DATA VS. REPEATED LABELING

One virtue of the previous algorithm is that it can seamlessly integrate the existence of "gold" data for learning the quality of labelers. It is a trivial change in the algorithm (i.e., do not update the true class of the "gold" examples, in Step 4 of Algorithm 1). We'll show next how the gold data can help to improve significantly the estimation accuracy in the case of sparse data.

5.1 Synthetic Experiments

The experimental setup is: we have a set of examples, equally assigned to two categories; a set of workers, with their quality (the sum of diagonal elements of the confusion matrix) ranges uniformly from 0.55 to 0.95. Since potential bias can be easily separated from error using our algorithm, we do not assume any biased workers here. Also, the qualities of workers are class-independent. We examine how accurately the algorithm can estimate:

- Average classification error
- Average cost estimation error

We conducted a set of synthetic experiments, in order to have the flexibillity of controlling the composition of the data set. We vary the fraction of gold data across four levels: 0% known examples, 25% known examples, 50% known examples, and 75% known examples. The three levels of data sparsity are 3 workers per example, 5 workers per example, and 10 workers per example, respectively.

The estimation results for average classification error are shown in Figure 1. As expected, the introduction of gold data indeed reduces the classification error to a certain degree. However, data sparsity acts as an important moderating variable here. When there are only 3 workers per example, we see a significant decrease on the classification errors. But with 10 workers per example, the gain is minimal, even with 75% gold data. An interesting finding is that, we can achieve the same performance by simply forcing workers to label more examples. When each worker can label 30 examples, the unsupervised algorithm works almost as well as the algorithm that uses 75% gold data, regardless of the data sparsity.

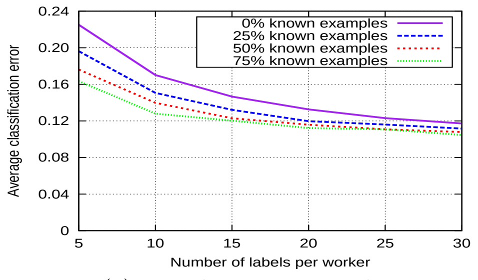
Figure 2 shows the results for average cost estimation error. The average cost estimation error for 100% known examples is displayed as a minimum bound for cost estimation. Similarly, data sparsity moderates the benefits of gold data. The improvement on cost estimation error is apparent when there are 3 workers per example, but negligible when there are 10 workers per example.

5.2 The Virtues of Gold Data

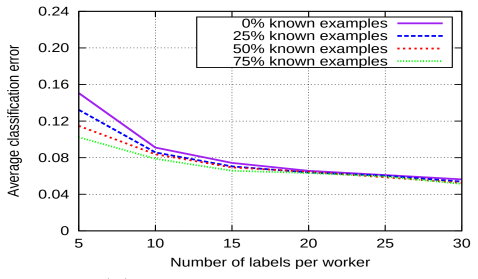
The experimental results above show us "the power of crowds". It seems that all the benefits that we gain from gold data can be realized by acquiring more labels from workers. However, several reasons make gold data still valuable in reality.

- **Worker attrition**: In real-life crowdsourcing systems, it is not always possible to force workers to label a large number of examples since workers might lose their patience for doing the same thing over and over again.

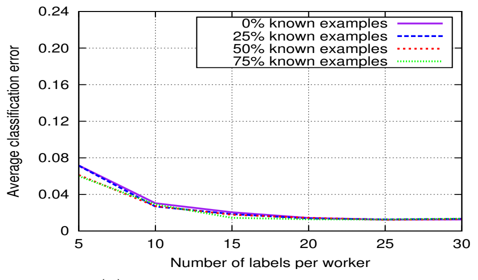
- **Imbalanced datasets**: For very imbalanced datasets, the probability that one example is from the minority class is extremely small. Therefore, we need a large number of unknown examples to ensure a relatively accurate estimate of worker quality, across all possible categories. The presence of gold data makes it possible to quickly test workers using data from all categories, rather than waiting for the occasional example from the minority category to appear.



(a) 3 workers per example



(b) 5 workers per example



(c) 10 workers per example

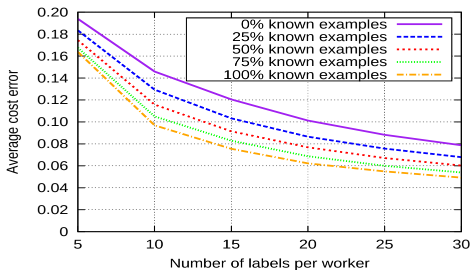
Figure 1: Average classification error

- **Low quality of workers**: AMT marketplace is a high-noise environment where low-quality workers like spammers are prevalent. Having a large number of such low-quality workers would make the estimation much more challenging. If we have some gold data in hand, we can get rid of this kind of workers quickly.

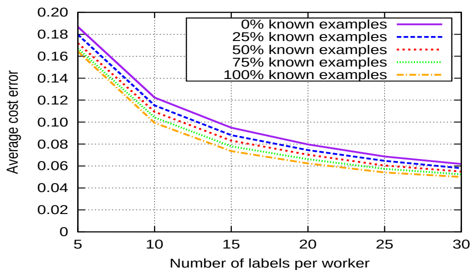
- **Calibrating results ("consistent bias")**: One assumption for the previous experiments is that there are all the workers have no systematic bias. But if we happen to have a set of workers that are biased in the same direction, we may end up with results that are consistent but biased. However, if the employer can provide a few gold data as anchor points, the estimation algorithms can adjust for the bias accordingly.

6. ACTIVE TESTING FOR WORKER QUALITY USING GOLD LABEL

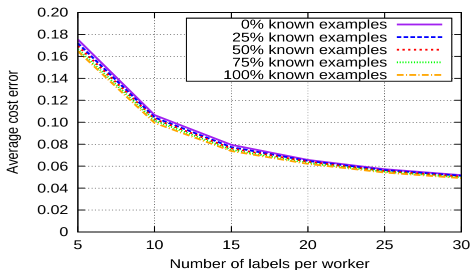
So far, we assumed that the number of examples that we use for testing a worker is a predetermined fraction of the total number of labels annotated by each worker. However, this is unnecessarily costly. As previous results demonstrate, the



(a) 3 workers per example



(b) 5 workers per example



(c) 10 workers per example

Figure 2: Average cost estimation error

addition of one more label has diminishing marginal return. However, we can always get some information from asking the worker to label a new example. Hence, there has to be a state where the gain that we expect from asking the worker to label a "gold" example, conditional on his expected lifetime, is no more than the gain from labeling a new example. In this section, we consider how to learn worker quality in a more active way. To make the illustration simple and clear, we start from the case of testing using gold label when we have only one worker.

The procedure goes as follows: When a worker comes, we first compute his expected cost based on his current "confusion matrix". Then, we use some techniques to predict his expected cost after giving him an additional gold example to label. Since we probably end up with different expected costs by feeding the worker with gold examples in different classes, we can choose the one which yields the lowest expected cost. If we have no label for an example, it is equivalent to have a label given by a *spammer* defined previously. The expected utilities from testing and labeling a new example are given below:

- **Testing**:

$$\begin{aligned}(ExpCost_{current} - \min ExpCost_{after\_testing})\\ \times LifeTime\_Remaining\end{aligned} \qquad (5)$$

- **Labeling a new example**:

$$(ExpCost_{spammer} - ExpCost_{current}) \qquad (6)$$

Having the expected utilities in hand, we can make our testing decisions accordingly: if the number in Equation 5 is larger than the number in Equation 6, we allocate a gold example in class that yields minimum expected cost after testing to the worker; otherwise, we give a new example to him.

So how can we predict the expected cost of the worker after testing? We explore this question next.

6.1 Dirichlet and Dirichlet Compound Multinomial Distribution

When presented with an example in class i, the probability vector of the assigned label follow a Dirichlet distribution $Dir(\alpha_i)$ where $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{iJ})$, given that the worker has classified the examples in class i into class j for $\alpha_{ij} - 1$ times in the past. So the quality of a worker can be fully captured by a set of Dirichlet distributions. Now suppose that we are going to test the quality of this worker for examples whose true class is i by feeding him n_i examples in class i. Then, the probability of samples x_i (the worker would classify the testing examples into each class), given the parameter vector α_i, follows a Dirichlet compound multinomial distribution. The probability can be calculated by the following explicit formula:

$$Pr(x_i|\alpha_i) = \frac{n_i!}{\prod_k (n_{ik})!} \frac{\Gamma(\sum_k \alpha_{ik})}{\Gamma(n_i + \sum_k \alpha_{ik})} \prod_k \frac{\Gamma(n_{ik} + \alpha ik)}{\Gamma(\alpha_{ik})}$$

where Γ represents the gamma function, n_{ik} is the number of times that the assigned label is k.

6.2 Expected Worker Cost over Time

Clearly, the quality of each worker can be fully captured by a set of Dirichlet distributions, one for each class. Also, we can predict the future outcomes after a number of tests by a set of Dirichlet compound multinomial distribution. So, the question becomes: can we reduce the expected worker cost over time by testing? If so, to what extent?

To answer this question, we conducted a synthetic experiment where have $J = 3$ classes. We assume uniform prior for the worker quality, which is $[(Dir(1,1,1)Dir(1,1,1)Dir(1,1,1))]'$. We examine the change of expected cost for the worker by varying the number of testing examples from 0 to 10 in each class (For simplicity, we make the number of testing examples the same across classes here).

Figure 3 illustrates how the expected cost of the worker decreases as we increase the number of testing examples. Not surprisingly, we see a diminishing marginal reduction for the cost. However, since it is likely that the classification decision given the assigned label will not change between two or more iterations, this is not the case for minimized cost.

6.3 Optimal Allocation of Testing Examples Across Categories

The previous experiments assume that the number of testing examples is given for each category. Here, we talk about how to optimally allocate testing examples based on three factors: prior class distribution, cost matrix, and Dirichlet priors. We assume the total number of testing examples $N = 10$. A brute force strategy is used here: we exhaustively test on all possible combinations and find the one that yields the lowest expected cost after testing. The size of the solid balls represent the expected cost after testing. So, the smaller the ball is, the more benefit we gain from testing. The numbers in the text box show the optimal allocation strategy.

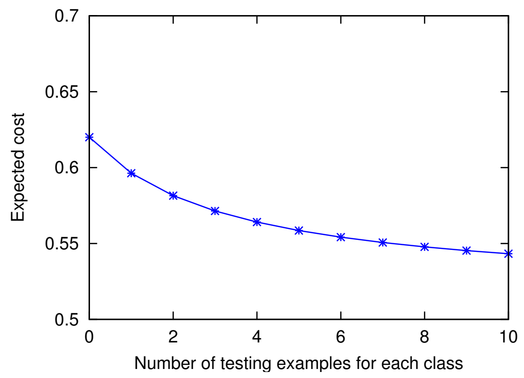


Figure 3: Expected cost of the worker varies with the number of testing examples

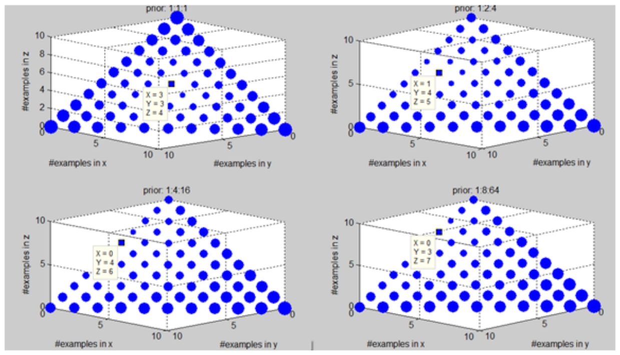


Figure 4: Optimal allocation of testing examples under different prior class distributions

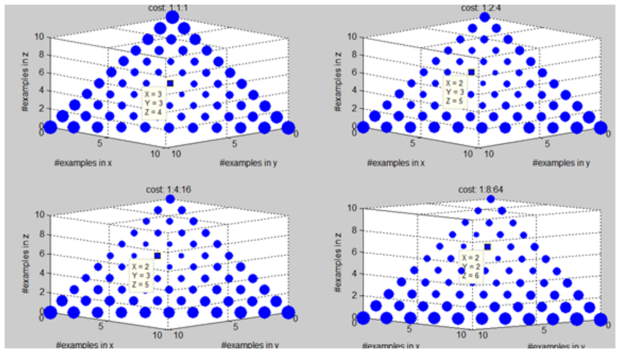


Figure 5: Optimal allocation of testing examples under different cost matrices

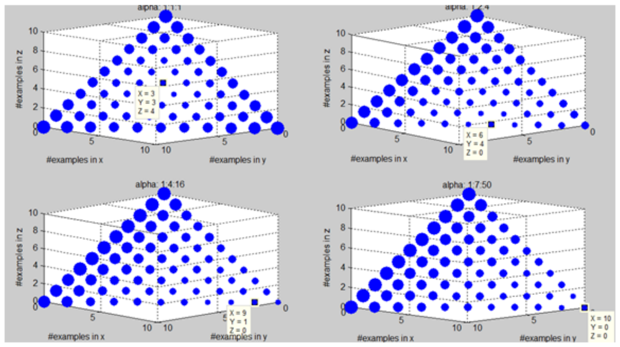


Figure 6: Optimal allocation of testing examples under different Dirichlet priors

Figure 4 demonstrates how the variation in prior class distribution would influence the optimal allocation decision. A straightforward conclusion is: all else being equal, we should allocate more testing examples for the majority class. Figure 5 shows that all else being equal, we should allocate more testing examples for classes with higher misclassification costs. From Figure 6, we conclude that all else being equal, we should allocate more testing examples for classes with smaller Dirichlet priors.

7. TESTING STRATEGIES IN SINGLE LABELING

The previous results illustrate the possible reduction in expected worker cost that we might gain from testing by gold data. To simply the problem, we examine the case of single labeling. In the scenario of single labeling, the expected benefit from testing a worker would be the reduction in worker cost, times the expected number of labels he would give in future (min{remaining lifetime, remaining number of examples per worker}). However, when we test the worker with a gold example, we forgo the opportunity to ask this worker to label a new example. The expected benefit from labeling a new example would be the expected cost with no information minus the expected cost of having a label assigned by this worker. When a worker comes, we need to decide whether to test him or give him a new example to label. We use the following experiments to test the performance of different testing strategies.

7.1 Experimental Setup

In our simulated experiment, we have 100 workers with varying degree of qualities, each of which has infinite lifetime. Therefore, the remaining number of examples per worker is a proxy of the expected number of labels he would give in future. We have three categories, whose prior probabilities are 0.2, 0.3, and 0.5, respectively. We have a cost of 1 when an object is misclassified, and 0 otherwise. We assume that worker accuracy is class independent and represents the value of diagonal element in confusion matrix. Also, the workers have equal probability of classifying example in one category to another two categories.

- low quality, low variance: worker accuracies are uniformly distributed in [0.35,0.7]
- high quality, low variance: worker accuracies are uniformly distributed in [0.65,1.0]
- high variance: worker accuracies are uniformly distributed in [0.35,1.0]

Our experiment goes as follows: we have a fixed budget which is used to pay for workers. To simplify the problem, we assume for now that a large number of gold examples are available for us to test on the quality of workers, so it cost us almost nothing to test a worker, except that we lose the opportunity to get another label for a new example from him. We conduct the experiment under four different budget levels, which allow us to pay for 4000, 6000, 8000, 10000 HITs, respectively.

7.2 Testing Strategies

- **No Testing**: When a worker comes, we always give him a new example to label. Since we invest no effort for testing, we have no information on worker quality.
- **Uniform Testing**: When a worker comes, we test him with a predefined probability using a gold example, the true class of which is randomly chosen.
- **Adaptive Testing**: When a worker comes, we test him with a predefined probability using a gold example. However, rather than random selection, we choose a gold example from the class which yields highest reduction in worker cost.
- **Active Testing**: When a worker comes, if we predict a higher benefit from testing, we give the worker another gold example to label; if not, we simply assign the worker a new example to label.

We compare different testing strategies across three dimensions: number of new examples labeled, average actual loss, and average estimated loss. The actual loss is the true cost we incur if we assign the example to the class which minimizes our expected cost, while the estimated loss is this minimum cost we expect, regardless of the true class. Both actual loss and estimated loss are computed only for newly labeled examples.

Example 5. *Consider a worker A that label web sites into porn and notporn. Suppose that when he is given a new site and he labels it as notporn: We compute the corresponding "soft" label (e.g.* $(0.6, 0.4)$*). The estimated loss of this label is* 0.4 *since we will assign the example into porn, but there is still* 40% *probability that the true class is notporn. If the true class is porn, then the actual loss will be* 0*; otherwise, the actual loss will be* 1. □

7.3 Results

The experimental results are shown in Figure 7. The x-axis represents the average actual loss, and the y-axis is the average estimated loss. The numbers alongside the arrows have the form m/n, where m is the number of new examples labeled and n is the number of HITs completed.

For **No Testing** strategy, the number of examples labeled will always equal to the number of HITs completed. Since we know nothing about the underlying worker quality, the only thing we can do is to believe what workers said and treat them as perfect workers. So, the estimated loss will always be zero. Therefore, the only non-trivial information is the actual loss we have. We make this strategy as a baseline by putting a vertical dashed line to indicate the average actual loss incurred under different conditions.

A perfect strategy for estimating worker cost will generate points laying on the diagonal line. Comparing it with the results given by different testing strategies, it is not difficult to find that we tend to overestimate the worker cost if worker quality is high, and underestimate the cost if worker quality is low. This is not surprising since we only use a small number of gold examples for testing.

There are several important messages implied in the Figure 7. First, the more we test, the better knowledge we have for the workers, and the lower average loss we'll suffer. Second, we get highest gain from testing when there are significant quality variance across workers. Third, as worker quality gets high, **Active Testing** would tend to label more new examples and use less gold examples. Fourth, **Active Testing** beats **Uniform Testing**, using the same number of gold example. The advantage of **Active Testing** is that it knows not only where to allocate the resources, but also how many resouces should be allocated.

Why are **Active Testing** so successful? One reason is that we can adjust for the potential bias of workers. Also, since we tend to test low-quality a lot more, we suppress the potential bad workers from participation. Therefore, the workers who label new examples are likely to be high-quality workers.

7.4 What if We can Block Bad Workers?

The results presented above show how much can we gain in data quality if we account for the varying quality of workers. However, we can do a lot more with the knowledge of worker quality. Nowadays, most crowdsourcing interfaces (e.g. AMT) allow requesters to block low-performance workers to discourage opportunistic behaviors. In order to see the effect of blocking, we modify our experiments by adding the following rule: if both of the two conditions hold, we will block the worker.

- The expected cost of a worker is higher than 0.9 times the expected cost of a spammer (i.e.$QualityScore < 0.1$).
- We have tested this worker a sufficient number of times (e.g. $5J^2$, where J is the number of classes) and have a relatively high confidence of blocking.

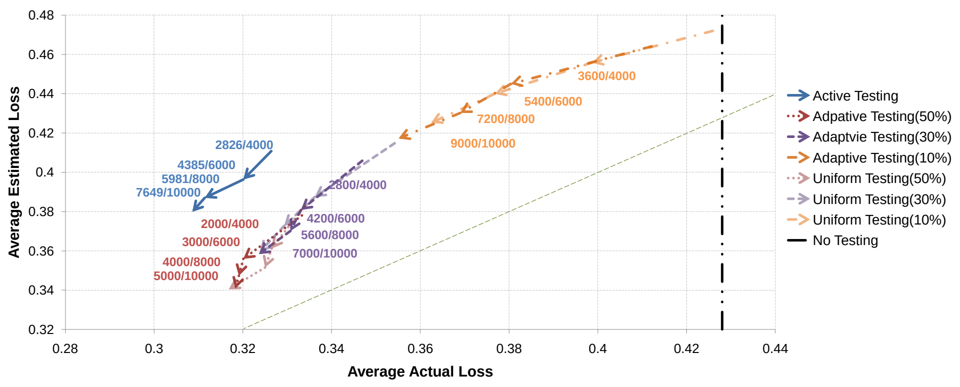
Table 1 show the results before and after the inclusion of rejection. Here, we only show the results for the case where the number of HITs allowed is 1000. As expected, if we can block bad workers, both the average true loss and estimated loss would be lower, and we can label a larger number of new examples. These results are pretty encouraging. Although we did not show the evidence here, in reality, we can also give bonus to high-performance workers to increase their propensity to participate, as long as we have a relatively accurate estimate for worker quality.

8. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

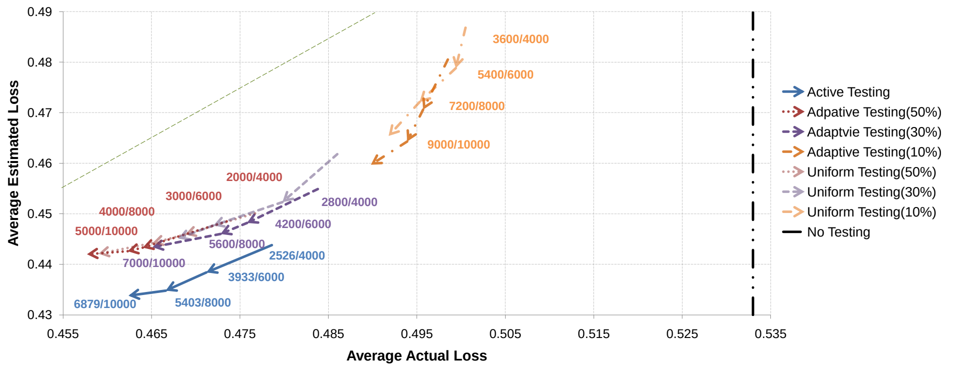
Repeated labeling has become a promising strategy since the advent of online crowdsourcing systems. In this paper, we first showed how to use repeated labeling not only for identifying the correct answer to each task, but also for measuring the labeling quality of each worker. Moreover, we put forward an algorithm to differentiate between low-quality workers and high-quality, but biased workers. Our algorithm generates a scalar score which represents the classification cost of each worker. We also illustrated how to seamlessly integrate unsupervised and supervised techniques for inferring the quality of the workers and test on their evaluation performances. Next, we brought up a strategy to actively test the quality of a worker. The experimental results show that our strategy performs significantly better than the traditional passive testing strategies.

Despite its contributions, this paper still has several limitations, which might undermine its practical usefulness.

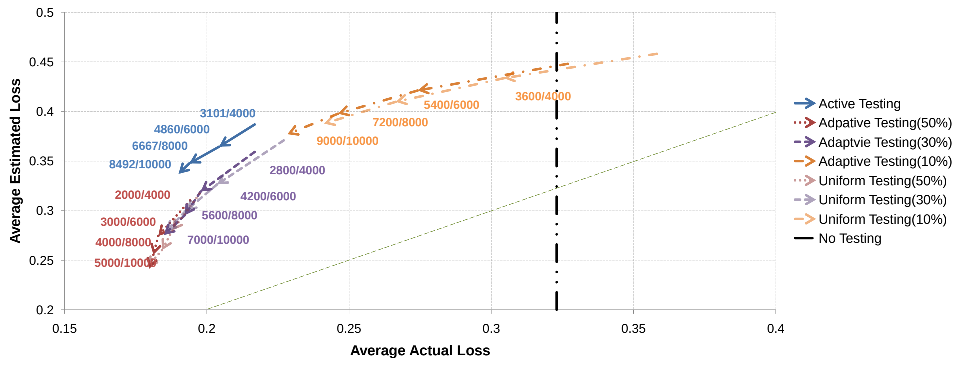
- For this study, we assume that the inherent qualities of workers are independent. In practice, workers might have correlated errors, either positively or negatively, which would certainly affect the validity of our algorithm. However, not all correlations are harmful. Previous research [3] has show that negative correlation between workers could increase the accuracy of classification results.



(a) Quality Distribution: U[0.35,1]



(b) Quality Distribution: U[0.35,0.7]



(c) Quality Distribution: U[0.65,1]

Figure 7: The average actual loss and estimated loss for uniform testing, adaptive testing, and active testing under different quality distributions

Quality Condition	Block Allowed	Average True Loss	Average Estimated Loss	# Examples Labeled	#Workers Blocked
high variance	No	0.3086	0.3799	7649	0
high variance	Yes	0.2848	0.3652	7834	18
low quality	No	0.4624	0.4338	6879	0
low quality	Yes	0.4366	0.4056	7065	41
high qualtiy	No	0.1900	0.3371	8492	0
high quality	Yes	0.1891	0.3373	8495	0

Table 1: Active Testing With and Without Rejection of Workers

- We assume that the error-rate of a worker is constant across different examples. Intuitively, we might expect that some examples are more difficult to label than others. Wallace et al. [7] developed a novel strategy that relies on the participating novice labelers to indicate which examples they are likely to mislabel, and use the experienced workers to label more difficult instances. We did not show how to estimate the example-conditional error-rates in our current paper.

- Our *active testing* strategy is only for the case when there is only one worker per example. This, of course, ignores the power of multiple workers. The case of multiple workers would complicate the problem to a large degree, for the reason that the benefit we gain from one worker also depends on who else has labeled a particular example.

- In this study, we assume workers have deterministic lifetime. In reality, we might expect the lifetime of workers are stochastic. For example, we can assume the lifetime of a worker follows geometric or long-tailed distribution.

The list above is probably incomplete. There still is much to do to come up with a better way to actively learn the qualities of workers. We hope that this study would inspire more valuable research in this area.

The code is open source and available at http://code.google.com/p/get-another-label/ and a demo is publicly accessible at http://qmturk.appspot.com/.

9. ACKNOWLEDGMENTS

Panagiotis G. Ipeirotis and Jing Wang were supported by the National Science Foundation under Grant No. IIS-0643846.

Bibliography

[1] Carpenter, B. Multilevel Bayesian models of categorical data annotation. Available at http://lingpipe-blog.com/lingpipe-white-papers/, 2008.

[2] Dawid, A. P., and Skene, A. M. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics 28*, 1 (Sept. 1979), 20–28.

[3] Kuncheva, L. I., Whitaker, C. J., Shipp, C. A., and Duin, R. P. W. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications 6*, 1 (2003), 22–31.

[4] Malone, T. W., Laubacher, R., and Dellarocas, C. Harnessing crowds: Mapping the genome of collective intelligence. Available at http://ssrn.com/abstract=1381502, 2010.

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

[6] Sheng, V. S., Provost, F., and Ipeirotis, P. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2008)* (2008).

[7] Wallacel, B. C., Small, K., Brodley, C. E., and Trikalinos, T. A. Who should label what? Instance allocation in multiple expert active learning. In *Proceedings of the Eleventh SIAM International Conference on Data Mining* (2011).

[8] Welinder, P., Branson, S., Belongie, S., and Perona, P. The multidimensionalwisdom of crowds. In *Advances in Neural Information Processing Systems 23 (NIPS 2010)* (2010).

[9] Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., and Movellan, J. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)* (2009).