OPERATIONS RESEARCH CENTER

Working Paper

Wisely Using a Budget for Crowdsourcing

by

Seyda Ertekin
Haym Hirsh
Cynthia Rudin

OR 392-12 April 2012

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Wisely Using a Budget for Crowdsourcing

Şeyda Ertekin SEYDA@MIT.EDU
MIT Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Haym Hirsh HIRSH@CS.RUTGERS.EDU
Department of Computer Science
Rutgers University,
Piscataway, NJ 08854, USA

Cynthia Rudin RUDIN@MIT.EDU
MIT Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

The problem of "approximating the crowd" is that of estimating the crowd's majority opinion by querying only a subset of it. Algorithms that approximate the crowd can intelligently stretch a limited budget for a crowdsourcing task. We present an algorithm, "CrowdSense," that works in an online fashion where examples come one at a time. CrowdSense dynamically samples subsets of labelers based on an exploration/exploitation criterion. The algorithm produces a weighted combination of a subset of the labelers' votes that approximates the crowd's opinion. We then introduce two variations of CrowdSense that make various distributional assumptions to handle distinct crowd characteristics. In particular, the first algorithm makes a statistical independence assumption of the probabilities for large crowds, whereas the second algorithm finds a lower bound on how often the current sub-crowd agrees with the crowd majority vote. Our experiments on CrowdSense and several baselines demonstrate that we can reliably approximate the entire crowd's vote by collecting opinions from a representative subset of the crowd.

Keywords: Crowdsourcing, Wisdom of crowds, Labeler quality estimation, Approximating the crowd, Aggregating the opinions

1. Introduction

Our goal is to determine the majority opinion of a crowd, on a series of questions ("examples"), with a limited budget. For example, let us consider a sneaker company, wanting to do a customer survey, where they provide free variants of a new sneaker to the members of a large crowd of product testers, who each give a "yay" or "nay" to each product, one at a time. The company wants to know whether the majority of testers likes each new sneaker variant. Some product testers are more useful than others; some align closely with the majority vote, and some do not. If the company can identify who are the most useful product testers, they can send new trial sneakers mostly to them, at a large cost savings.

This problem of estimating the majority vote on a budget goes well beyond product testing – the problem occurs for many tasks falling under the umbrella of *crowdsourcing*, where the collective intelligence of large crowds is leveraged by combining their input on a set of examples. One of the most commonly known crowdsourced databases was used in the Netflix Contest[1]. This database contains 100 million anonymous movie ratings, and could definitely be used to predict the crowd's opinion on movies, even though it was not explicitly designed for that purpose.

In product design and manufacturing, companies have taken steps to interact with their customers and having them suggest, discuss and vote on new product ideas (Ogawa and Piller, 2006; Sullivan, 2010). They also commonly rely on focus groups and usability studies to collect the opinions of crowds on existing or new products. In the absence of sophisticated sampling techniques, collecting more votes per item increases the likelihood that the crowd's opinion reflects the majority opinion of the entire population.

In cases where each vote is provided at a cost, collecting a vote from every member of the crowd in order to determine the majority opinion can be expensive and may not even be attainable under fixed budget constraints. Because of the open nature of crowdsourcing systems, it is not necessarily easy to approximate the majority vote of a crowd on a budget by sampling a representative subset of the voters. For example, the crowd may be comprised of labelers with a range of capabilities, motives, knowledge, views, personalities, etc. Without any prior information about the characteristics of the labelers, a small sample of votes is not guaranteed to align with the true majority opinion. In order to effectively approximate the crowd, we need to determine who are the most representative members of the crowd, in that they can best represent the interests of the crowd majority. This is even more difficult to accomplish when items arrive over time, and it requires our budget to be used both for 1) estimating the majority vote, even before we understand the various qualities of each labeler, and 2) exploring the various labelers until we can estimate their qualities well. Estimating the majority vote in particular can be expensive before the labelers' qualities are known, and we do not want to pay for additional votes that are not likely to impact the decision.

In order to make economical use of our budget, we could determine when just enough votes have been gathered to confidently align our decision with the crowd majority. The budget limitation necessitates a compromise: if we pay for a lot of votes per decision, our estimates will closely align with the crowd majority, but we will only make a smaller number of decisions, whereas if we pay for less votes per decision, the accuracy may suffer, but more decisions are made. There is clearly an exploration/exploitation tradeoff: before we can exploit by using mainly the best labelers, we need to explore to determine who these labelers are, based on their agreement with the crowd.

The main contributions of this work can be broken down into two main parts: In the first part, we propose a modular algorithm, *CrowdSense*, that approximates the wisdom of the crowd. In an online fashion, CrowdSense dynamically samples a subset of labelers, determines whether it has enough votes to make a decision, and requests more if the decision is sufficiently uncertain. CrowdSense keeps a balance between exploration and exploitation in its online iterations – exploitation in terms of seeking labels from the highest-rated

1. http://www.netflixprize.com

labelers, and exploration so that enough data are obtained about each labeler to ensure that we learn each labeler's accuracy sufficiently well.

The second part presents probabilistic variations of CrowdSense. One of the main challenges to approximating the crowd has to do with the fact that the majority vote is taken as the ground truth (the truth we aim to predict). This means that there is a complicated relationship (a joint probability distribution) between the labelers' accuracies and the ground truth. In Sections 7.1 and 7.2 of the paper, we introduce two variations of CrowdSense, called CrowdSense.Ind and CrowdSense.Bin, that make various distributional assumptions to handle distinct crowd characteristics. In particular, the first algorithm makes a statistical independence assumption of the probabilities for large crowds, whereas the second algorithm finds a lower bound on how often the current sub-crowd agrees with the crowd majority vote, using binomial distribution. For both CrowdSense.Ind and CrowdSense.Bin, even though probabilistic assumptions were made, the accuracy is comparable to (or lower than) CrowdSense itself. It is difficult to characterize the joint distribution for the problem of approximating the crowd, due to the constraint that the majority vote is the true label. CrowdSense, with its easy-to-understand weighted majority voting scheme, seems to capture the essence of the problem, and yet has the best performance within the pool of algorithms we tried, so our experiments mainly focus on CrowdSense.

Throughout the paper, a "majority vote" refers to the simple, every day sense of voting wherein every vote is equal, with no differential weighting of the votes. This is in contrast to a weighted majority vote, as we use in CrowdSense, wherein each labeler's vote is multiplied by the labeler's quality estimate. This weighting scheme ensures that the algorithm places a higher emphasis on the votes of higher quality labelers.

The rest of the paper is organized as follows: The next section presents related work. In Section 3, we present details on CrowdSense within the general context of a modular framework for approximating the wisdom of the crowd. An outline of the datasets that we used and the baselines we considered are presented in Section 4. In Section 5, we demonstrate the performance of CrowdSense against the baselines, and present the impact of specific choices within the general framework. Section 6 outlines the impact of various initialization and estimation criteria on CrowdSense's performance. Section 7 presents details on CrowdSense.Ind and CrowdSense.Bin and presents empirical evaluation of these algorithms. The paper concludes with final remarks and directions for future work. In Appendix A, we present a proposition for CrowdSense.Bin that states that the computed scores are nonnegative. Appendix B presents the pseudocode of a "flipping" technique – a heuristic for taking the opposite vote of labelers that have quality estimates less than 0.5 – for CrowdSense.Ind, and shows the comparison of the prediction accuracy of CrowdSense.Ind with and without labeler flipping in Figure 10.

2. Related Work

The low cost of crowdsourcing labor has increasingly led to the use of resources such as Amazon Mechanical Turk[2] (AMT) to label data for machine learning purposes, where collecting multiple labels from non-expert annotators can yield results that rival those of experts. This cost-effective way of generating labeled collections using AMT has also been used in several

2. http://www.mturk.com

studies (Nakov, 2008; Snow et al., 2008; Sorokin and Forsyth, 2008; Kaisser and Lowe, 2008; Dakka and Ipeirotis, 2008; Nowak and Rüger, 2010; Bernstein et al., 2010, 2011). While crowdsourcing clearly is highly effective for easy tasks that require little to no training of the labelers, the rate of disagreement among labelers has been shown to increase with task difficulty (Sorokin and Forsyth, 2008; Gillick and Liu, 2010), labeler expertise (Hsueh et al., 2009) and demographics (Downs et al., 2010). Regardless of the difficulty of the task or their level of expertise, offering better financial incentives does not improve the reliability of the labelers (Mason and Watts, 2009; Marge et al., 2010), so there is a need to identify the level of expertise of the labelers, to determine how much we should trust their judgment.

Dawid and Skene (1979) presented a methodology to estimate the error rates based on the results from multiple diagnostic tests without a gold standard using latent variable models. Smyth et al. (1994a,b) used a similar approach to investigate the benefits of repeatedly labeling same data points via a probabilistic framework that models a learning scheme from uncertain labels. Although a range of approaches are being developed to manage the varying reliability of crowdsourced labor (see, for example Law and von Ahn, 2011; Quinn and Bederson, 2011; Callison-Burch and Dredze, 2010; Wallace et al., 2011), the most common method for labeling data via the crowd is to obtain multiple labels for each item from different labelers and treat the *majority label* as an item's true label. Sheng et al. (2008), for example, demonstrated that repeated labeling can be preferable to single labeling in the presence of label noise, especially when the cost of data preprocessing is non-negligible. Dekel and Shamir (2009a) proposed a methodology to identify low quality annotators for the purpose of limiting their impact on the final attribution of labels to examples. To that effect, their model identifies each labeler as being either good or bad, where good annotators assign labels based on the marginal distribution of the true label conditioned on the instance and bad annotators provide malicious answers. Dekel and Shamir (2009b) proposed an algorithm for pruning the labels of less reliable labelers in order to improve the accuracy of the majority vote of labelers. First collecting labels from labelers and then discarding the lower quality ones presents a different viewpoint than our work, where we achieve the same "pruning effect" by estimating the qualities of the labelers and not asking the low quality ones to vote in the first place. A number of researchers have explored approaches for learning how much to trust different labelers, typically by comparing each labeler's predictions to the majority-vote prediction of the full set. These approaches often use methods to learn both labeler quality characteristics and latent variables representing the *ground-truth* labels of items that are available (e.g. Kasneci et al., 2011; Warfield et al., 2004; Dekel et al., 2010), sometimes in tandem with learning values for other latent variables such as task difficulty (Whitehill et al. (2009); Welinder et al. (2010)), classifier parameters (Yan et al., 2010a,b; Raykar et al., 2010), or domain-specific information about the labeling task (Welinder et al., 2010).

Our work appears similar to the preceding efforts in that we similarly seek predictions from multiple labelers on a collection of items, and seek to understand how to assign weights to them based on their prediction quality. However, previous work on this topic viewed labelers mainly as a resource to use in order to lower uncertainty about the true labels of the data. In our work, we could always obtain the true labels by collecting all labelers' votes and determining their majority vote. We seek to approximate the correct prediction at lower cost by decreasing the number of labelers used, as opposed to increasing accuracy by turning to additional labelers at additional cost. In other words, usually we do not know the classifier and try to learn it, whereas here we know the classifier and are trying to approximate it. This work further differs from most of the preceding efforts in that they presume that learning takes place after obtaining a collection of data, whereas our method also works in online settings, where it simultaneously processes a stream of arriving data while learning the different quality estimates for the labelers. Sheng et al. (2008) is one exception, performing active learning by reasoning about the value of seeking additional labels on data given the data obtained thus far. Donmez et al. (2009) propose an algorithm, IEThresh, to simultaneously estimate labeler accuracies and train a classifier using labelers' votes to actively select the next example for labeling. Zheng et al. (2010) present a two-phase approach where the first phase is labeler quality estimation and identification of high quality labelers, and the second phase is the selection of a subset of labelers that yields the best cost/accuracy tradeoff. The final prediction of the subset of labelers is determined based on their simple majority vote. We discuss the approach taken by Donmez et al. (2009) further in Section 4 as one of the baselines to which we compare our results, because this is the only algorithm we know of aside from ours that can be directly applied to approximating the crowd in the online setting. IEThresh uses a technique from reinforcement learning and bandit problems, where exploration is done via an *upper confidence bound* on the quality of each labeler. CrowdSense's quality estimates are instead Bayesian shrinkage estimates of the labelers' qualities.

3. CrowdSense

Let us first model the labelers' quality estimates as a measure of their agreement with the crowd majority. These quality estimates indicate whether a labeler is "representative" of the crowd.

Let $L = \{l_1, l_2, \ldots, l_M\}$, $l_k : \mathcal{X} \to \{-1, 1\}$ denote the set of labelers and $\{x_1, x_2, \ldots, x_t, \ldots, x_N\}$, $x_t \in \mathcal{X}$ denote the sequence of examples, which could arrive one at a time. We define $V_{it} := l_i(x_t)$ as l_i's vote on x_t and $S_t \subset \{1, \ldots, M\}$ as the set of labelers selected to label x_t. For each labeler l_i, we then define c_{it} as the number of times we have observed a label from l_i so far:

$$c_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{[i \in S_{\tilde{t}}]} \tag{1}$$

and define a_{it} as how many of those labels were consistent with the other labelers:

$$a_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{\left[i \in S_{\tilde{t}}, V_{i\tilde{t}} = V_{S_{\tilde{t}}\tilde{t}}\right]} \tag{2}$$

where $V_{S_t t} = \text{sign}\left(\sum_{i \in S_t} V_{it} Q_{it}\right)$ is the weighted majority vote of the labelers in S_t. Labeler l_i's quality estimate is then defined as

$$Q_{it} := \frac{a_{it} + K}{c_{it} + 2K} \tag{3}$$

where t is the number of examples that we have collected labels for and K is a smoothing parameter. Q_{it} is a Bayesian shrinkage estimate of the probability that labeler i will agree with the crowd, pulling values down toward 1/2 when there are not enough data to get a more accurate estimate. This ensures that labelers who have seen fewer examples are not considered more valuable than labelers who have seen more examples and whose performance is more certain. CrowdSense uses a pessimistic estimate of the mean rather than an upper (e.g. 95%) confidence interval that IEThresh (Donmez et al., 2009) (and UCB algorithms) use.

At the beginning of an online iteration to label a new example, the labeler pool is initialized with three labelers; we select two "exploitation" labelers that have the highest quality estimates Q_{it} and select another one uniformly at random for "exploration". This initial pool of seed labelers enables the algorithm to maintain a balance between exploitation of quality estimates and exploration of the quality of the entire set of labelers. We ask each of these 3 labelers to vote on the example. The votes obtained from these labelers for this example are then used to generate a *confidence score*, given as

$$\text{Score}(S_t) = \sum_{i \in S_t} V_{it} Q_{it}$$

which represents the weighted majority vote of the labelers. Next, we determine whether we are certain that the sign of $\text{Score}(S_t)$ reflects the crowd's majority vote, and if we are not sure, we repeatedly ask another labeler to vote on this example until we are sufficiently certain about the label. To measure how certain we are, we greedily see whether adding an additional labeler might make us uncertain about the decision. Specifically, we select the labeler with the highest quality estimate Q_{it}, who is not already in S_t, as a candidate to label this example. We then check whether this labeler could potentially either change the weighted majority vote if his vote were included, or if his vote could bring us into the *regime of uncertainty* where the $\text{Score}(S_t)$ is close to zero, and the vote is approximately a tie. We check this before we pay for this labeler's vote, by temporarily assigning him a vote that is opposite to the current subcrowd's majority. The criteria for adding the candidate labeler to S_t is defined as:

$$\frac{|\text{Score}(S_t)| - Q_{l_{\text{candidate}},t}}{|S_t| + 1} < \varepsilon \tag{4}$$

where ε controls the level of uncertainty we are willing to permit, $0 < \varepsilon \leq 1$. If (4) is true, the candidate labeler is added to S_t and we get (pay for) this labeler's vote for x_t. We then recompute $\text{Score}(S_t)$ and follow the same steps for the next-highest-quality candidate from the pool of unselected labelers. If the candidate labeler is not added to S_t, we are done adding labelers for example t, and assign the weighted majority vote as the predicted label of this example. We then proceed to label the next example in the collection. Pseudocode for the CrowdSense algorithm is given in Algorithm 1.

4. Datasets and Baselines

We conducted experiments on various synthetic and real-world datasets that model a crowd from two separate perspectives; the first perspective models the crowd in the traditional sense where the crowd comprises of human labelers, whereas the second perspective models the crowd in terms of competing predictive models and the goal is to approximate the

Algorithm 1 Pseudocode for CrowdSense.

1. **Input:** Examples $\{x_1, x_2, \ldots, x_N\}$, Labelers $\{l_1, l_2, \ldots, l_M\}$, confidence threshold ε, smoothing parameter K.

2. **Define:** $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates.

3. **Initialize:** $a_{i1} \leftarrow 0,\ c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$. Initialize $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$

4. **For** $t = 1, ..., N$

   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K},\ i = 1, \ldots, M$. Update L_Q.

   (b) $S_t = \{l^{(1)}, l^{(2)}, l^{(k)}\}$, where k is chosen uniformly at random from the set $\{3, \ldots M\}$.

   (c) **For** $j = 3 \ldots M,\ j \neq k$

      i. $\text{Score}(S_t) = \sum_{i \in S_t} V_{it} Q_{it},\quad l_{\text{candidate}} = l^{(j)}$.

      ii. If $\frac{|\text{Score}(S_t)| - Q_{l_{\text{candidate}},t}}{|S_t|+1} < \varepsilon$, then $S_t \leftarrow S_t \cup l_{\text{candidate}}$. Otherwise exit loop to stop adding new labelers to S_t.

   (d) Get the weighted majority vote of the labelers $V_{S_t t} = \text{sign}\left(\sum_{i \in S_t} V_{it} Q_{it}\right)$

   (e) $\forall i \in S_t$ where $V_{it} = V_{S_t t},\quad a_{it} \leftarrow a_{it} + 1$

   (f) $\forall i \in S_t,\quad c_{it} \leftarrow c_{it} + 1$

5. **End**

common prediction of these models. Computational model results can be expensive, both in the sense of cost and time (e.g. large-scale simulations).

MovieLens is a movie recommendation dataset of user ratings on a collection of movies, and the goal is to find the majority vote of these reviewers. The dataset is originally very sparse, meaning that only a small subset of users have rated each movie. We compiled a smaller subset of this dataset where each movie is rated by each user in the subset, to enable comparative experiments, where labels can be "requested" on demand. We mapped the original rating scale of [0-5] to votes of {-1,1} by using 2.5 as the threshold. ChemIR is a dataset of chemical patent documents from the 2009 TREC Chemistry Track. This track defines a "Prior Art Search" task, where the competition is to develop algorithms that, for a given set of patents, retrieve other patents that they consider relevant to those patents. The evaluation criteria is based on whether there is an overlap of the original citations of patents and the patents retrieved by the algorithm. The ChemIR dataset that we compiled is the complete list of citations of several chemistry patents, and the {+1,-1} votes indicate whether or not an algorithm has successfully retrieved a true citation of a patent. In our dataset, we do not consider false positives; that is, the predicted citations that are not in the original citations are not counted as -1 votes. Both MovieLens and ChemIR datasets have 11 labelers in total. Reuters is a popular dataset of articles that appeared on

MovieLens		ChemIR		Reuters		Adult	
48.17(L1)	54.01(L8)	50.72(L1)	78.62(L8)	80.76(L1)	80.21(L8)	81.22(L1)	85.51(L8)
89.78(L2)	47.44(L9)	46.78(L2)	82.06(L9)	83.00(L2)	78.68(L9)	80.59(L2)	81.32(L9)
93.43(L3)	94.16(L10)	84.46(L3)	50.12(L10)	89.70(L3)	95.06(L10)	86.22(L3)	85.54(L10)
48.90(L4)	95.62(L11)	88.41(L4)	50.98(L11)	82.98(L4)	82.88(L11)	87.63(L4)	79.74(L11)
59.12(L5)		86.69(L5)		88.12(L5)	71.57(L12)	91.12(L5)	84.86(L12)
96.35(L6)		87.46(L6)		87.04(L6)	87.54(L13)	94.11(L6)	96.71(L13)
87.59(L7)		49.52(L7)		95.42(L7)		56.68(L7)	

Table 1: The number of examples in each dataset and the true accuracies of the labelers.

the Reuters newswire in 1987. We selected documents from the *money-fx* category. The Reuters data is divided into a "training set" and a "test set," which is not the format we need to test our algorithms. We used the first half of the training set (3,885 examples) to develop our labelers. Specifically, we trained several machine learning algorithms on these data: AdaBoost, Naïve Bayes, SVM, Decision Trees, and Logistic Regression, where we used several different parameter settings for SVM and Decision Trees. Each algorithm with its specific parameter setting is used to generate one labeler and there were 10 labelers generated this way. Additionally, we selected 3 features of the dataset as labelers, for a total of 13 labelers. We combined the other half of the training set (omitting the labels) with the test set, which provided 6,904 total examples over which we used to measure the performance of CrowdSense and the baselines. The same simulation of a crowd that we conducted for the Reuters dataset was also used for the Adult dataset from the UCI Machine Learning Repository, which is collected from the 1994 Census database, where we aim to predict whether a person's annual income exceeds $50K.

For MovieLens, we added 50% noise and for ChemIR we added 60% noise to 5 of the labelers to introduce a greater diversity of judgments. This is because all the original labelers had comparable qualities and did not strongly reflect the diversity of labelers and other issues that we aim to address. For the Reuters and Adult datasets, we varied the parameters of the algorithms' labelers, which are formed from classification algorithms, to yield predictions with varying peformance. All reported results are averages of 100 runs, each with a random ordering of examples to prevent bias due to the order in which examples are presented.

We compared CrowdSense with several baselines: (a) the accuracy of the *average* labeler, represented as the mean accuracy of the individual labelers, (b) the accuracy of the overall best labeler in hindsight, and (c) the algorithm that selects just over half the labelers (*i.e.* $\lceil 11/2 \rceil = 6$ for ChemIR and MovieLens, $\lceil 13/2 \rceil = 7$ for Reuters and Adult) uniformly at random, which combines the votes of labelers with no quality assessment using a majority vote. We also compare CrowdSense against IEThresh (Donmez et al., 2009). IEThresh builds upon Interval Estimation (IE) learning, and estimates an upper confidence interval UI for the mean reward for an action, which is a technique used in reinforcement learning. In IEThresh, an action refers to asking a labeler to vote on an item, and a reward represents the labeler's agreement with the majority vote. The UI metric for IEThresh is defined for a sample "a" as:

$$UI(a) = m(a) + t_{\frac{\alpha}{2}}^{(n-1)} \frac{s(a)}{\sqrt{n}} \tag{5}$$

where $m(a)$ and $s(a)$ are the sample mean and standard deviation for a, n is the sample size observed from a and $t_{\frac{\alpha}{2}}^{(n-1)}$ is the critical value for the Student's t-distribution. The sample "a" for a labeler is the vector of ± 1's indicating agreement of that labeler with the majority. IEThresh updates the UI scores of the labelers after observing new votes, and given $\{UI_1, UI_2, \ldots, UI_k\}$ for the labelers, IEThresh selects all labelers with $UI_{\tilde{j}} > \varepsilon \times \max_{\tilde{j}}(UI_{\tilde{j}})$. The ε parameter in both CrowdSense and IEThresh algorithms tunes the size of the subset of labelers selected for voting, so we report results for a range of ε values. Note that tuning ε exhibits opposite behavior in CrowdSense and IEThresh; increasing ε relaxes CrowdSense's selection criteria to ask for votes from more labelers, whereas larger ε causes IEThresh to have a more strict selection policy. So a given value of ε for CrowdSense does not directly correspond to a particular value of ε for IEThresh. On the other hand, since ε controls the number of labelers used for each example in both algorithms, it also controls the total number of labelers used for the entire collection of examples. (This is proportional to the cost of the full experiment.) When we adjust the ε values for CrowdSense and IEThresh so that the total number of labelers is similar, we can directly see which algorithm is more accurate, given that comparable total cost is spent on each.

It is worth noting that we do not assess the performance of the algorithms on separate test splits of the datasets. Rather, we make a single pass over the *entire* dataset and select labelers for each example based on the quality estimates available at that particular time. This is different than how IEThresh was evaluated in (Donmez et al., 2009), where the labelers' qualities were first learned on a training set, and then the single best labeler with the highest accuracy was selected to label all the examples in a separate test set. In order to have a valid comparative assessment of the iterative nature of quality estimation, the majority vote for each example in IEThresh is computed based on the majority vote of the selected sub-crowd, similar to CrowdSense.

4.1 A Type of Baseline that Does Not Work Well – "Learning the Weights"

We experimented with algorithms where the weights λ_j are *learned*, or determined implicitly, using machine learning algorithms. To do this, we used the data and labels collected so far in order to fit the λ_j's and produce predictions. We describe one algorithm, which learns weights via AdaBoost (Freund and Schapire, 1997), and uses CrowdSense's mechanism for adding labelers. AdaBoost constructs a classifier as a linear combination of "weak" classifiers. In the context of approximating the majority vote, each weak classifier (also known as a *feature*) corresponds to a labeler and their classifications are the observed labels from that labeler. In formal terms, let $\mathcal{F}$ denote the hypothesis space that is the class of convex combinations of features $\{h_j\}_{j=1\ldots n}$, where $h_j : \mathcal{X} \to \{-1, 1\}$. The function $f \in \mathcal{F}$ is then defined as a linear combination of the features:

$$f := f_{\boldsymbol{\lambda}} := \sum_j \lambda_j h_j,$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ are the minimizers of the objective function. Consider we have already collected votes on t examples, and are selecting labelers for the next example. Following the initialization scheme of CrowdSense, we initialize with three labelers that follow the exploration/exploitation tradeoff and add new labelers on-demand based on exploitation qualities. We minimize AdaBoost's objective considering all the votes we have seen so far from the selected labelers and the next candidate labeler. Let λ_{S} denote the weights of the labelers already selected, and λ_c denote the candidate labeler's weight. We decide whether to add the candidate labeler to the set based on

$$\frac{|V_{t+1,S}\lambda_{\text{S}}| - |\lambda_c|}{\left(\sum_{i \in \text{S}} |\lambda_i|\right) + |\lambda_c|} < \varepsilon$$

where $V_{t+1,S}$ are the votes of the selected labelers. The normalization needs to depend on the λ's, since the scaling of λ's produced by AdaBoost can be extremely large. The criteria of adding the new labeler follows the same intuition as CrowdSense and determines whether the vote of the new labeler would bring us into the *regime of uncertainty*. Our experiments with this approach yielded poor results compared to CrowdSense.

The reason why this approach fails is because of overfitting in the early iterations. The amount of available data in early iterations is, by definition, small. This leads to problems with overfitting, which leads to poor predictions. This, in turn, leads to problems with the labels used in later iterations, which again leads to poor predictions in later iterations. The predictions become worse and worse, and the overall results were worse than many, if not all, of the baselines.

On the other hand, one could use a better method (like CrowdSense) to more accurately calculate the weights in early iterations, and then switch over to machine learning of the weights after a sufficiently large amount of data has been established.

5. Overall Performance

We compare CrowdSense with the baselines to demonstrate its ability to accurately approximate the crowd's vote. Accuracy is calculated as the proportion of examples where the algorithm agreed with the majority vote of the entire crowd.

Figure 1 indicates that uniformly across different values of ε, CrowdSense consistently achieved the highest accuracy against the baselines, indicating that CrowdSense uses any fixed budget more effectively than the baselines, including IEThresh. Generally, the quality estimates of CrowdSense better reflect the true accuracy of the members of the crowd and therefore, it can identify and pick a more representative subset of the crowd. The other baselines that we considered did not achieve the same level of performance as CrowdSense and IEThresh. On the subplots in Figure 1, the accuracy of the best labeler in hindsight (baseline (b)) is indicated as a straight line. Note that the accuracy of the best labeler is computed separately as a constant. Baselines (a) and (c), which are the average labeler and the unweighted random labelers, achieved performance beneath that of the best labeler. For the MovieLens dataset, the values for these baselines are 74.05% and 83.69% respectively; for ChemIR these values are 68.71% and 73.13%, for Reuters, the values are 84.84% and 95.25%, and for Adult they are 83.94% and 95.03%. The results demonstrate that asking for labels from labelers at random may yield poor performance for representing

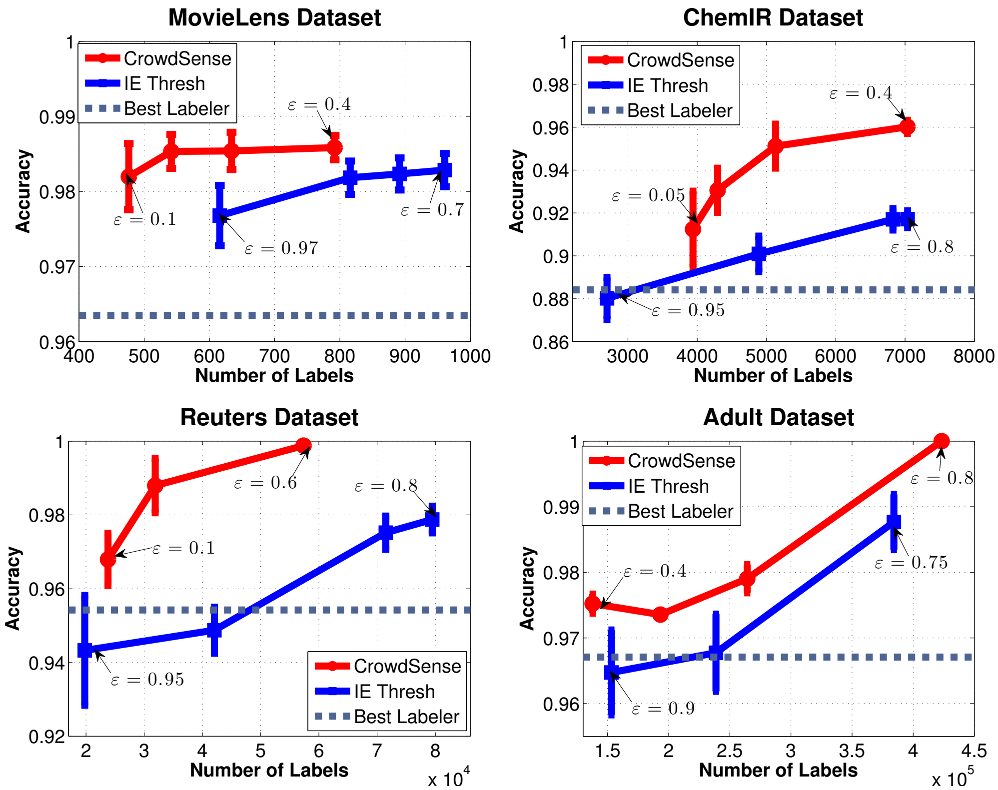


Figure 1: Tradeoff curves, averaged over 100 runs. The x-axis is the total number of votes (the total cost) used by the algorithm to label the entire dataset. The y-axis indicates the accuracy on the full dataset.

the majority vote, highlighting the importance of making informed decisions for selecting the representative members of the crowd. Asking the best and average labeler were also not effective approximators of the majority vote.

6. Specific Choices for Exploration and Exploitation in CrowdSense

The algorithm template underlying CrowdSense has three components that can be instantiated in different ways: (i) the composition of the initial seed set of labelers (step 4(b) in the pseudocode), (ii) how subsequent labelers are added to the set (step 4(c)), and (iii) the weighting scheme, that is, the quality estimates. This weighting scheme affects the selection of the initial labeler set, the way the additional labelers are incorporated, as well as the strategy for combining the votes of the labelers (steps 4(b)(c)(d)).

Our overall results of this section are the algorithm is fairly robust to components (i) and (ii), but not to component (iii): the weighting scheme is essential, but changing the composition of the seed set and the way subsequent labelers are added does not heavily affect accuracy. There is a good reason why CrowdSense is robust to changes in (i) and (ii). CrowdSense already has an implicit mechanism for exploration, namely the Bayesian

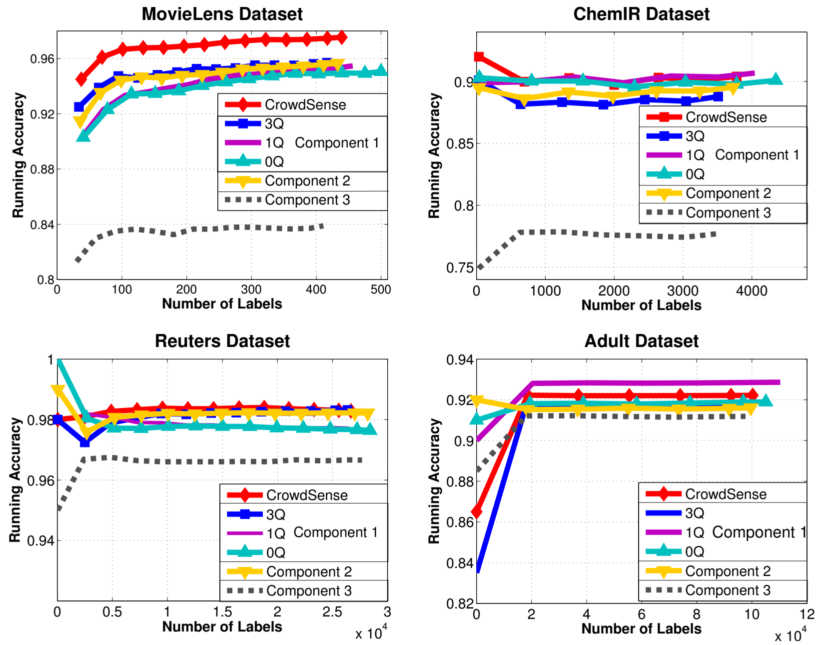


Figure 2: Performance of CrowdSense and several of its variants, averaged over 100 runs. The x-axis is the total number of votes (the total cost), and the y-axis is the running accuracy on those examples. For the plots, we used $\varepsilon = 0.005$ for MovieLens, $\varepsilon = 0.01$ for ChemIR, $\varepsilon = 0.2$ for Reuters, and $\varepsilon = 0.1$ for Adult dataset.

shrinkage estimates of quality, discussed in Section 6.3, so the exploration provided in (i) is not the only exploration mechanism. When adding new labelers in (ii), this is already after forming the initial set, including good labelers for exploitation. So exploitation in adding new labelers does not always make an additional impact. In other words, even if we reduce exploration or exploitation somewhere within CrowdSense, other means of exploration/exploitation can compensate, which makes CrowdSense's result fairly robust. We tested the effect of the first component (i) by running separate experiments that initialize the labeler set with three (3Q), one (1Q), and no (0Q) labelers that have the highest quality estimates, where for the latter two, additional labelers are selected at random to complete the set of three initial labelers. 3Q removes the exploration capability of the initial set whereas the latter two make limited use of the quality estimates.

Figure 2 shows the time course, averaged over 100 runs for a specific choice of ε for all three variants. The most important points in this figure are the final accuracy of final total number of labels (i.e. the budget). Note that each variant implements a separate labeler selection and/or label weighting scheme which, in turn, affects the total number of labels collected by each variant. Some of the datasets (Reuters, ChemIR) did not show much change. The MovieLens dataset, the smallest dataset in our experiments, shows the largest gain in the predictive performance of CrowdSense compared to the other three variants. This is also the dataset where the difference in the number of labels collected for each curve is reasonably comparable across the variants – so for the same total budget, the performance

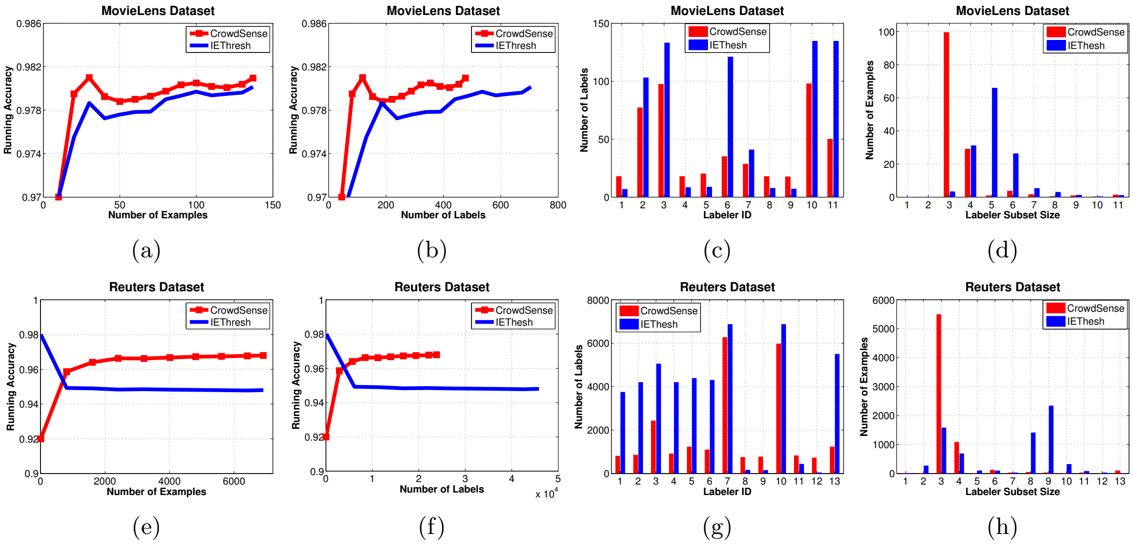


Figure 3: Comparison of CrowdSense and IEThresh on MovieLens and Reuters datasets, averaged over 100 runs. The first two columns represent the accuracy as a function of the number of examples seen and labels collected, respectively. The third column presents the number of votes collected from each labeler, and the last column shows the breakdown of the number of labelers selected to vote on each example.

of CrowdSense was better. For the Adult dataset, 1Q achieves the highest final accuracy, but it also collects close to 10K more labels than CrowdSense, so the accuracies are not directly comparable because the budget is different. This agrees with the compromise that we discussed in the Introduction, regarding the decisions on how to spend the budget on collecting labels.

We experimented with the second component (*ii*) by adding labelers randomly rather than in order of their qualities. In this case, exploitation is limited, and the algorithm again tends not to perform as well on most datasets (as shown in Figure 2 for the curve marked "Component 2") either in terms of accuracy or budget. To test the effect of the weighting scheme in the third component (*iii*), we removed the use of weights from the algorithm. This corresponds to selecting the initial seed of labelers and the additional labelers at random without using their quality estimates. In addition, when combining the votes of the individual labelers, we use majority voting, rather than a weighted majority vote. This approach performs the worst among all the variants in all four datasets, demonstrating the significance of using quality estimates for labeler selection and the calculation of the weighted vote.

In Figure 3, we compare CrowdSense and IEThresh on MovieLens and Reuters datasets from several perspectives to compare their runtime performance and the breakdown of their labeler choices. For both datasets, we used $\varepsilon = 0.1$ for CrowdSense and $\varepsilon = 0.9$ for IEThresh, which allows IEThresh to use approximately twice the budget of CrowdSense. The average running accuracy (over 100 runs) in plots (Figures 3(a)(e)) indicate

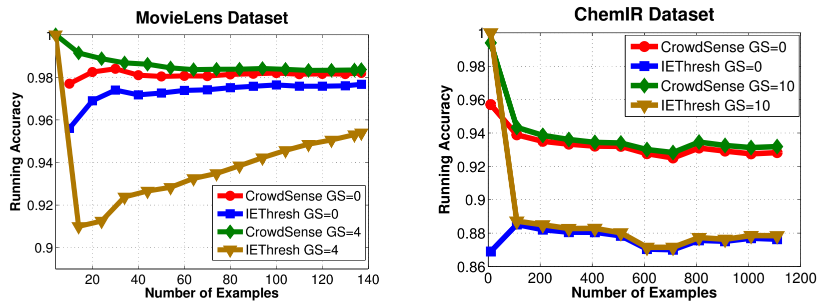


Figure 4: Comparison of running accuracy with and without gold standard, averaged over 100 runs. For CrowdSense, we used $\varepsilon = 0.1$ and for IEThresh, we used $\varepsilon = 0.97$ to have comparable number of labelers.

that CrowdSense achieves higher accuracy and collects fewer number of labels (Figures 3(b)(f)) than IEThresh, even with a much lower budget. Figures 3(c) and 3(g) present a bar plot of how many times each labeler was selected and Figures 3(d) and 3(h) show a bar plot of how many labelers were selected to vote on each round. For instance, for the MovieLens dataset, Figure 3(d) shows that CrowdSense most often chooses 3 labelers per examples whereas IEThresh rarely chooses labelers per example and prefers 4,5,6, or 7 labelers per example. This also indicates that CrowdSense makes better use of the fewer number of votes it collects. Both algorithms collect labels mostly from the highest accuracy labelers (see Table 4 for labeler accuracies), but IEThresh is less effective in combing these labels. IEThresh and CrowdSense both select the highest quality labelers, where IEThresh combines them by using a simple majority vote, while CrowdSense uses a weighted majority vote; an interesting observation is that CrowdSense's weighted majority vote (of fewer labelers) is more effective for approximating the crowd's true simple majority vote than IEThresh's simple majority vote (of a larger number of labelers) – this is, in some sense, ironic.

6.1 Initialization with Gold Standard

Gold standard examples are the "actual" opinion of the crowd gathered by taking the majority vote of all members. Even though collecting votes from the entire crowd for some of the examples initially increases the overall cost, it might help us make better estimate the true characteristics of the labelers earlier, and thus, achieve higher overall accuracy. In CrowdSense, initialization with gold standard refers to updating step 3 in the pseudocode by initializing c_{i1} with the number of gold standard examples and a_{i1} with the number of times labeler l_i agrees with the entire crowd for those gold standard examples.

In Figure 4, we present the effect of initializing with gold standard for CrowdSense and IEThresh. For the MovieLens dataset, the gold standard set contains four examples, where two of them were voted as +1 by the majority of the crowd, and the other two voted as -1. Since ChemIR is a larger dataset, we increase the gold standard set to 10 examples with an equal number of positives and negatives. The curves in Figure 4 start after observing the gold data.

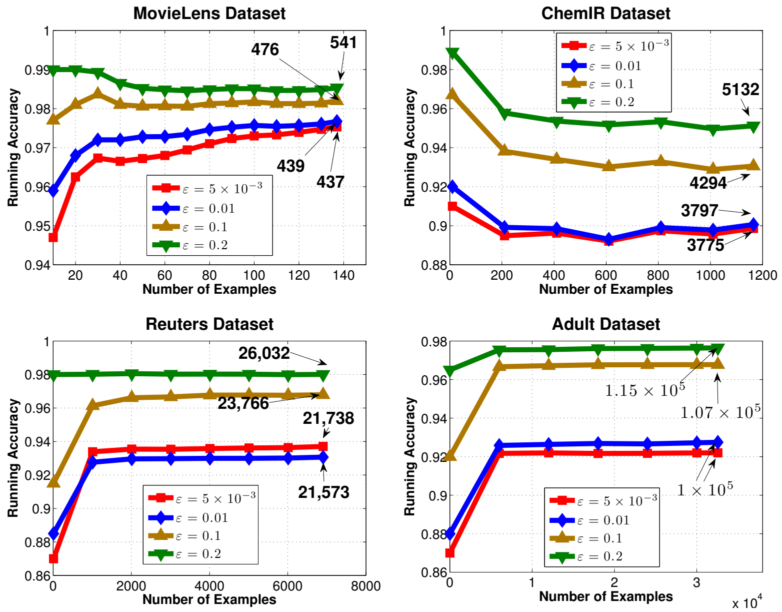


Figure 5: Comparison of the running accuracy of CrowdSense at various ε values, averaged over 100 runs. In the plots, we show how many total votes were requested at each ε value.

As expected, gold standard data clearly improves CrowdSense's performance because the estimates are now initialized with perfect information for the first few examples. An interesting observation is that providing gold standard data to IEThresh can actually make its performance substantially worse. Consider a labeler who agrees with the crowd's vote on every gold standard example. In this case, the labeler will get a reward for every single vote, yielding $UI = 1$ (since $s(a) = 0$ and $m(a) = 1$). On the other hand, the labelers that disagree with the crowd on some examples will have standard error $s(a) > 0$ due to missing rewards, so these labelers will receive $UI > 1$ and therefore they will be preferred over the labelers that were in total agreement with the entire crowd's vote for each gold standard example. Examples of this phenomenon are shown in Figure 4, where IEThresh performs worse with gold standard than without. This illustrates why we believe that upper confidence intervals may not be the right way to handle this particular problem. In any case, we note that it may be possible to achieve better performance for IEThresh by using the upper confidence interval for a multivariate binomial distribution rather than the t-distribution, since the number of correct votes is approximately binomially distributed rather than normally distributed.

6.2 Effect of the ε parameter

As discussed throughout Sections 4 and 5, the ε parameter is a trade-off between the total cost that we are willing to spend and the accuracy that we would like to achieve. Figure 5 illustrates the average running performance over 100 runs of CrowdSense for various values

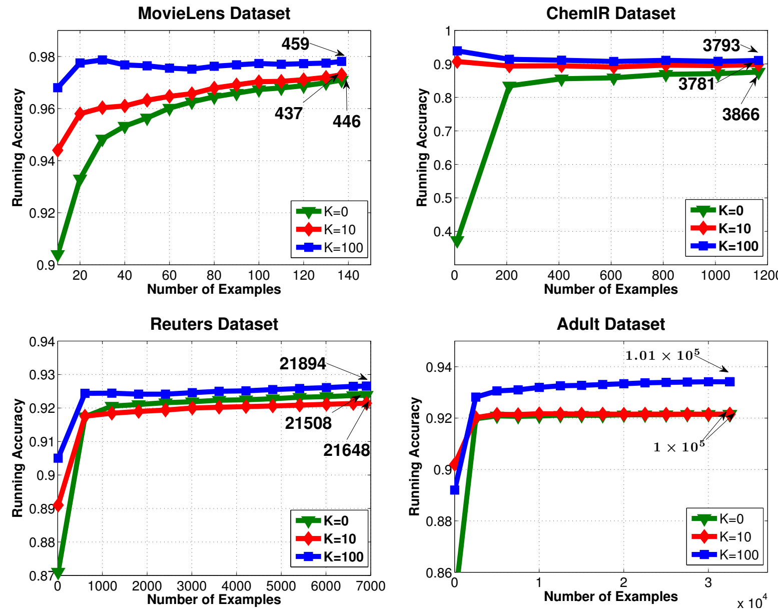


Figure 6: Comparison of running accuracy with varying K. All curves use $\varepsilon = 0.005$, and are averaged over 100 runs.

of epsilon. Each final point on each of the Figure 5 curves corresponds to a single point in Figure 1. Figure 5 shows that over the full time course, increasing epsilon leads to increased accuracy, at a higher cost.

6.3 Effect of the K parameter

The K parameter helps with both exploration at early stages and exploitation at later stages. In terms of exploration, K ensures that labelers who have seen fewer examples (smaller c_{it}) are not considered more valuable than labelers who have seen many examples. The quality estimate in (3) is a shrinkage estimator, lowering probabilities when there is uncertainty. Consider a labeler who was asked to vote on just one example and correctly voted. His vote should not be counted as highly as a voter who has seen 100 examples and correctly labeled 99 of them. This can be achieved using K sufficiently large.

Increasing K also makes the quality estimates more stable, which helps to permit exploration. Since the quality estimates are all approximately equal in early stages, the weighted majority vote becomes almost a simple majority vote. This prevents CrowdSense from trusting any labeler too much early on. Having the Q_{it}'s be almost equal also increases the chance to put CrowdSense into the "regime of uncertainty" where it requests more votes per example, allowing it to explore the labelers more.

We demonstrate the impact of K by comparing separate runs of CrowdSense with different K in Figure 6. Considering the MovieLens dataset, at $K = 100$, the quality estimates tilt the selection criteria towards an exploratory scheme in most of the iterations. At the other extreme, $K = 0$ causes the quality estimates to be highly sensitive to the accuracy of the labels right from the start. Furthermore, it is also interesting to note that removing K achieves worse performance than $K = 10$ despite collecting slightly more votes. This indicates that the results are better when more conservative quality estimates are used in early iterations.

7. Probabilistic Algorithms for Approximating the Crowd

CrowdSense has four important properties: 1) It takes labelers' votes into account even if they are right only half of the time. This property is especially beneficial for approximating small crowds, 2) It trusts "better" labelers more, and places higher emphasis on their votes, 3) Its decision rule is derived to be similar to boosting, and 4) Bayesian shrinkage of the quality estimates. These estimates provide the means for exploration of labelers in the earlier stages, and exploitation in later stages.

The fact that the majority vote determines the ground truth indicates a complicated relationship, a joint probability distribution, between the labelers' accuracies and the majority vote. CrowdSense makes an implicit assumption on the joint probability distribution through its boosting-style update. We believe it is possible to further improve its accuracy by making an explicit assumption on the joint probability distribution. To that effect, we propose two variations of CrowdSense that directly incorporate the joint probability distributions of the labelers under different assumptions. The first variation (CrowdSense.Ind) makes a statistical independence assumption of the labelers. That is, we will assume that the majority vote is independent of the votes that we have seen so far. This assumption is approximately true for large crowds, but does not hold for smaller crowds. This variation has a nice probabilistic interpretation, replacing the boosting-style update, to improve upon property 3. However, it sacrifices property 1. Therefore, its benefits are geared towards large crowds. The second variation (CrowdSense.Bin) makes a different probabilistic assumption, which is a lower bound on how often the current subcrowd agrees with the majority vote. This leads to a different voting scheme which is based on the binomial distribution of the votes. This also replaces the boosting-style decision rule in property 3. CrowdSense.Bin does not include the current subcrowd's weights in the vote, but the labeler selection criteria still favors labelers with high accuracy. Consequently, this approach covers the second property to some extent, but not as strong as the original CrowdSense.

7.1 CrowdSense.Ind

CrowdSense.Ind makes an assumption, namely that the quality of the individual labelers gives much more information than the count of votes received so far. In other words, that a labeler who is right 90% of the time and votes +1 should be counted more than a handful of mediocre labelers who vote -1. This assumption is approximately true when there a large number of labelers, but is not true for a small number of labelers. For instance, consider a case where we have received three votes so far: [+1,-1,-1], from labelers with qualities .8, .5, and .5 respectively. Let's say that there are a total of 500 labelers. The evidence suggests that the majority vote will be +1, in accordance with the high quality voter, and discounting the low-quality labelers. This is the type of assumption CrowdSense.Ind makes. Instead, if there were only 5 labelers total, evidence suggests that the majority vote might be -1, since the low quality labelers still contribute to the majority vote, and only one more -1 vote is now needed to get a majority.

Let V_i denote the vote of labeler i on a particular example, and P_i denote the probability that the labeler will agree with the crowd's majority vote. Consider that two labelers have voted 1 and -1, and we want to evaluate the probability that the majority vote is 1. Then, from Bayes Rule, we have

$$
\begin{aligned}
P(y=1|V_1=1,V_2=-1) &= \frac{P(V_1=1,V_2=-1,y=1)}{P(V_1=1,V_2=-1)} \\
P(V_1=1,V_2=-1,y=1) &= P(y=1)P(V_1=1,V_2=-1|y=1) \\
&= P(y=1)P(V_1=V_{maj},\, V_2 \neq V_{maj}) \\
&= P(y=1)P_1(1-P_2) \\
P(V_1=1,V_2=-1) &= P(y=1)P(V_1=1,V_2=-1|y=1) \\
&\quad + P(y=-1)P(V_1=1,V_2=-1|y=-1) \\
P(y=1|V_1=1,V_2=-1) &= \frac{P_1(1-P_2)P(y=1)}{P(y=1)P_1(1-P_2)+P(y=-1)(1-P_1)P_2}
\end{aligned}
$$

where $P(y=1)$ and $P(y=-1)$ are the ratios of the crowd's approximated votes of 1 and -1 on the examples voted so far, respectively.

To expand this more generally to arbitrary size subsets of crowds, we define the following notation: Let V_i^{bin} denote the mapping of labeler i's vote from $\{-1,1\}$ to $\{0,1\}$, i.e. $V_i^{\text{bin}} = (V_i+1)/2$. Using the following notation for the joint probabilities of agreement and disagreement with the crowd:

$$
\begin{aligned}
\psi_i &= P_i^{V_i^{\text{bin}}}(1-P_i)^{1-V_i^{\text{bin}}} && \text{(probability of agreement given vote } V^{\text{bin}}) \\
\theta_i &= (1-P_i)^{V_i^{\text{bin}}}P_i^{1-V_i^{\text{bin}}} && \text{(probability of disagreement given vote } V^{\text{bin}})
\end{aligned}
$$

the likelihood of the majority vote y being 1 is estimated by the following conditional probability:

$$
\begin{aligned}
f(x_t|\text{votes}) = P(y=1|\underbrace{V_1,V_2,\ldots,V_i}_{\text{votes of labelers in } S}) &= \frac{\left(\prod_{l\in S}\psi_l\right)P_+}{\left(\prod_{l\in S}\psi_l\right)P_+ + \left(\prod_{l\in S}\theta_l\right)(1-P_+)} \\
&= \frac{1}{1+\dfrac{\left(\prod_{l\in S}\theta_l\right)(1-P_+)}{\left(\prod_{l\in S}\psi_l\right)P_+}}
\end{aligned}
\tag{6}
$$

where probabilities higher than 0.5 indicate a majority vote estimate of 1, and -1 otherwise. Further, the more the probability in (6) diverges from 0.5, the more we are confident of the current approximation of the crowd based on the votes seen so far.

Note that labelers who are accurate less than half the time can be "flipped" so the opposite of their votes are used. That is, observing a vote V_i with $P_i < 0.5$ is equivalent to observing $-V_i$ with probability $1-P_i$. In Section 9, we discuss about this flipping heuristics and present graphs that demonstrate the effect of this approach.

As in CrowdSense, the decision of whether or not to get a vote from an additional labeler depends on whether his vote could bring us to the regime of uncertainty. As in CrowdSense,

Algorithm 2 Pseudocode for CrowdSense.Ind.

1. **Input:** Examples $\{x_1, x_2, \ldots, x_N\}$, Labelers $\{l_1, l_2, \ldots, l_M\}$, confidence threshold ε, smoothing parameter K.

2. **Define:** $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates. P_+ is the probability of a +1 majority vote.

3. **Initialize:** $a_i \leftarrow 0,\ c_i \leftarrow 0$ for $i = 1, \ldots, M$. Initialize $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$

4. **Loop for** $t = 1, ..., N$

   (a) $Q_{it} = \frac{a_i+K}{c_i+2K}, \quad \forall i$

   (b) Update L_Q with the new quality estimates.

   (c) Select 3 labelers and get their votes. $S_t = \{l^{(1)}, l^{(2)}, l^{(k)}\}$, where k is chosen uniformly at random from the set $\{3, \ldots, M\}$.

   (d) $V_{it}^{\text{bin}} = \frac{(V_{it}+1)}{2}, \quad \forall i \in S_t$

   (e) $\psi_{it} = Q_{it}^{V_{it}^{\text{bin}}}(1 - Q_{it})^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$

   (f) $\theta_{it} = (1 - Q_{it})^{V_{it}^{\text{bin}}} Q_{it}^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$

   (g) **Loop for** candidate $= 3 \ldots M$, candidate $\neq k$

      i. $f(x_t|\text{votes}) = \dfrac{1}{1 + \dfrac{(\prod_{i \in S_t} \theta_{it})(1 - P_+)}{(\prod_{i \in S_t} \psi_{it}) P_+}}$

      ii. $\psi_{\text{candidate}} = 1 - Q_{\text{candidate},t}$

      iii. $\theta_{\text{candidate}} = Q_{\text{candidate},t}$

      iv. $f(x_t|\text{votes}, V_{\text{candidate},t}) = \dfrac{1}{1 + \dfrac{(\prod_{i \in S_t} \theta_{it})\theta_{\text{candidate}}(1 - P_+)}{(\prod_{i \in S_t} \psi_{it})\psi_{\text{candidate}} P_+}}$

      v. **If** $f(x_t|\text{votes}) \geq 0.5$ **and** $f(x_t|\text{votes}, V_{\text{candidate},t}) < 0.5 + \varepsilon$ **then** ShouldBranch = 1

      vi. **If** $f(x_t|\text{votes}) < 0.5$ **and** $f(x_t|\text{votes}, V_{\text{candidate},t}) > 0.5 - \varepsilon$ **then** ShouldBranch = 1

      vii. **If** ShouldBranch = 1 **then** $S_t = \{S_t \cup \text{candidate}\}$, get the candidate's vote. **else** Don't need more labelers, break out of loop.

   (h) $\hat{y}_t = 2 \times \mathbb{1}_{[f(x_t|\text{votes})>0.5]} - 1$

   (i) $a_i = a_i + 1, \quad \forall i \in S_t$ where $V_{it} = \hat{y}_t$

   (j) $c_i = c_i + 1, \quad \forall i \in S_t$

5. **End**

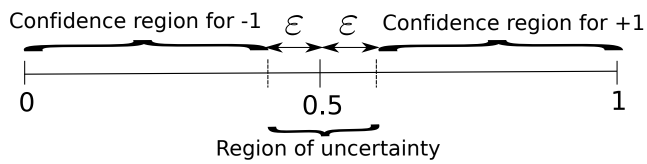


Figure 7: The confidence intervals of the probabilistic approach with the independence assumption of the labelers. The ε neighborhood of the decision boundary is the region of uncertainty.

this corresponds to hypothetically getting a vote from the candidate labeler that disagrees with the current majority vote. This corresponds to hypothetically getting a -1 vote when $P(y = 1|\text{votes}) > 0.5$, and getting a 1 vote otherwise. Defining

$$\psi_{\text{candidate}} = \begin{cases} 1 - P_{\text{candidate}} & f(x_t|\text{votes}) > 0.5 \\ P_{\text{candidate}} & \text{otherwise} \end{cases}$$

$$\theta_{\text{candidate}} = \begin{cases} P_{\text{candidate}} & f(x_t|\text{votes}) > 0.5 \\ 1 - P_{\text{candidate}} & \text{otherwise} \end{cases}$$

and expanding (6) to include the new hypothetical vote from the candidate labeler, we get

$$f(x_t|\text{votes}, V_{\text{candidate}}) = \frac{1}{1 + \frac{\left(\prod_{l \in S} \theta_l\right) \theta_{\text{candidate}} (1 - P_+)}{\left(\prod_{l \in S} \psi_l\right) \psi_{\text{candidate}} P_+}}. \tag{7}$$

The decision as to whether get a new vote depends on the values of Eq (6) and (7). If we are sufficiently confident of the current majority vote to the point where the next best labeler we have could not change our view, or is not in the regime of uncertainty (shown in Figure 7), then we simply make a decision and do not call any more votes. Pseudocode of CrowdSense.Ind is in Algorithm 2.

7.2 CrowdSense.Bin

The statistical independence assumption that we made in Section 7.1 is almost valid for sufficiently large crowds, but it does not hold when the crowd is small. In this subsection, we modify our treatment of the joint probability distribution. Specifically, we estimate a lower bound on how often the sub-crowd S_t agrees with the crowd's majority vote.

Consider again the scenario where there are 5 labelers in the crowd, and we already have observed [+1, -1, -1] votes from three labelers. The simple majority of the crowd would be determined by 3 votes in agreement, and we already have two votes of -1. So the problem becomes estimating the likelihood of getting one more vote of -1 from the two labelers that have not voted yet, which is sufficient to determine the simple majority vote. If the voters are independent, this can be determined from a computation on the binomial distribution. Let N_{needed} denote the number of votes needed for a simple majority and let N_{unvoted} denote the number of labelers that have not yet voted. (In the example above with 5 labelers, $N_{\text{needed}} = 1$ and $N_{\text{unvoted}} = 2$). In order to find a lower bound, we consider what

Algorithm 3 Pseudocode for CrowdSense.Bin

1. **Input:** Examples $\{x_1, x_2, \ldots, x_N\}$, Labelers $\{l_1, l_2, \ldots, l_M\}$, confidence threshold ε, smoothing parameter K.

2. **Define:** $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates.

3. **Initialize:** $a_{i1} \leftarrow 0,\ c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$.

4. **For** $t = 1, ..., N$

   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K},\ i = 1, \ldots, M$. Update L_Q.

   (b) $S_t = \{l^{(1)}, l^{(2)}, l^{(k)}\}$, where k is randomly sampled from the set $\{3, \ldots, M\}$.

   (c) **For** candidate $= 3 \ldots M,\ j \neq k$

      i. $l_+ := \sum_{S_t} \mathbb{1}_{[V_{l_t}=1]}$

      ii. $l_- := \sum_{S_t} \mathbb{1}_{[V_{l_t}=-1]}$

      iii. $l_{\text{current\_maj}} = \max(l_+, l_-)$

      iv. Majority_Is $= \lceil \frac{M}{2} \rceil$

      v. Num_Needed_For_Majority = Majority_Is - $l_{\text{current\_maj}}$

      vi. Score $= \left[\sum_{X=N_{\text{needed}}}^{N_{\text{unvoted}}} B(X, N_{\text{unvoted}}, 0.5)\right] - 0.5$

      vii. **If** Score $< \varepsilon$ **then** $S_t = S_t \cup l^{(\text{candidate})}$ **Else** Don't need more labelers, break out of loop

   (d) **End**

   (e) $\hat{y} = 1 - 2\mathbb{1}_{[|l_-|>|l_+|]}$

   (f) $\forall i \in S_t$ where $V_{it} = \hat{y},\quad a_{it} \leftarrow a_{it} + 1$

   (g) $\forall i \in S_t,\quad c_{it} \leftarrow c_{it} + 1$

5. **End**

happens if the remaining labelers have the worst possible accuracy, P=0.5. We then define the score using the probability of getting enough votes from the remaining crowd that agree with the current majority vote:

$$\begin{aligned}\text{Score} &= P_{X\sim\text{Bin}(\cdot,N_{\text{unvoted}},0.5)}(X \geq N_{\text{needed}}) - 0.5 \\ &= \left[\sum_{X=N_{\text{needed}}}^{N_{\text{unvoted}}} \text{Bin}(X, N_{\text{unvoted}}, 0.5)\right] - 0.5, \end{aligned} \tag{8}$$

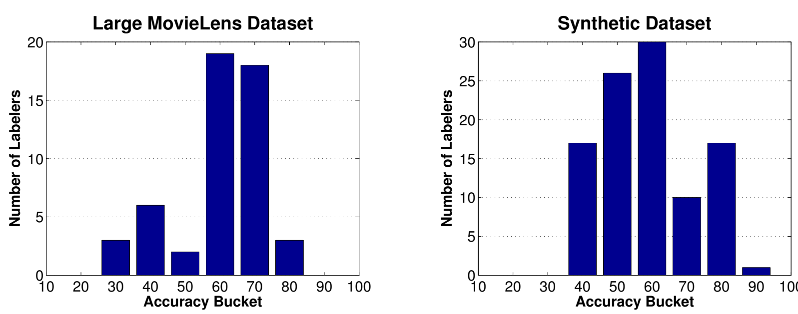
where $\text{Bin}(X, N_{\text{unvoted}}, 0.5)$ is the X^{th} entry in the Binomial distribution with parameters N_{unvoted} and probability of success of 0.5. In other words, this score represents a lower bound on how confident we are that the breakdown of votes that we have observed so far is consistent with the majority vote. The score (8) is always nonnegative; this is shown in

Proposition 1 in appendix A. Therefore, the decision to ask for a new vote can then be tied to our level of confidence, and if it drops below a certain threshold ε, we can ask for the vote of the highest quality labeler that has not voted yet. The algorithm stops asking for additional votes once we are sufficiently confident that the subset of labels that we have observed is a good approximation of the crowd's vote. The pseudocode of this approach is given in Algorithm 3.

8. Experiments

We generated two new datasets that simulate large crowds. The first dataset contains 2000 examples and 101 synthetically generated labelers; a precedent for using synthetically generated "radiologist" labelers is provided in Raykar et al. (2010). The majority votes of the examples are +1 for 1122 examples, and -1 for the remaining 878 examples. The accuracies of the labelers are distributed over the 43-90% range. The second dataset extends the original MovieLens dataset that we described earlier. Due to the sparsity of the dataset, we randomly grouped labelers in sets of five and aggregated the votes of the labelers in each group. That is, for each movie that was rated by at least one labeler in a group, we took the average of the ratings for that movie to represent the vote of that group. This way, we were able to extend the dataset to 51 groups and 175 examples. The positive to negative ratio of the examples is 100/75, and after adding 75% noise to the votes of the worst 11 groups, the qualities ranged from 37% to 81%. For uniformity, we will refer to each labeler group in the expanded MovieLens dataset simply as labelers of the dataset. The breakdown of the labelers' accuracies are presented in Figure 8.

Figure 9 shows the tradeoff curves for CrowdSense, CrowdSense.Ind, and CrowdSense.Bin. On most datasets, CrowdSense achieves better performance than CrowdSense.Ind and CrowdSense.Bin, though CrowdSense.Bin was the best performer on the Large MovieLens dataset, and CrowdSense.Ind was marginally better than CrowdSense on the large Synthetic dataset.



(a) N=175, M=51

(b) N=2000, M=101

Figure 8: The breakdown of labeler accuracies over 10% intervals for the datasets, used in our experiments. N and M denote the number of examples and the labelers in each dataset, respectively.

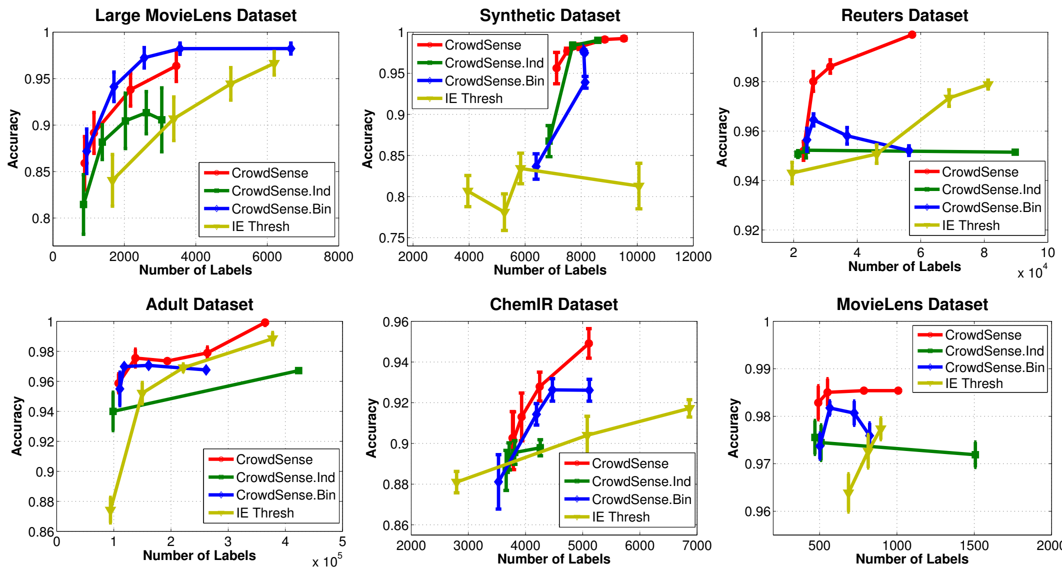


Figure 9: Tradeoff curves for all datasets at comparable ε values. We used $K = 10$ for CrowdSense and variants, and initialized all algorithms with 4 examples as the gold standard data.

9. Flipping Labelers

Labelers who have a negative correlation with the majority vote can be "flipped" to consider the opposite of their votes. We will demonstrate how to do this using CrowdSense.Ind. When a labeler is flipped, observing a vote V_i with $P_i < 0.5$ is equivalent to observing $-V_i$ with probability $1 - P_i$. In the online iterations, if a labeler's quality estimate drops below 0.5, we flip the labeler's original vote and use this vote as the labeler's non-adversarial vote. For these labelers' quality estimates, their a coefficients are incremented when they *disagree* with the majority vote. In Algorithm 4 of Appendix B, we present the pseudocode of CrowdSense.Ind with flipping, and compare the prediction accuracy of CrowdSense.Ind with and without labeler flipping in Figure 10.

10. Conclusions

Our goal is to "approximate the crowd," that is, to estimate the crowd's majority vote by asking only certain members of it to vote. We discussed exploration/exploitation in this context, specifically, exploration for estimating the qualities of the labelers, and exploitation (that is, repeatedly using the best labelers) for obtaining a good estimate of the crowd's majority vote. We presented a modular outline that CrowdSense follows, which is that a small pool of labelers vote initially, then labelers are added incrementally until the estimate of the majority is sufficiently certain, then a weighted majority vote is taken as the overall prediction. We discussed specific choices within this modular outline, the most important one being the choice of ε, which determines the overall budget for labeling the entire dataset.

We compared our results to several baselines, indicating that CrowdSense, and the overall exploration/exploitation ideas behind it, can be useful for approximating the crowd.

We then presented two variations of CrowdSense, namely CrowdSense.Ind, which is based on an independence assumption, and CrowdSense.Bin, where calculations are based on the binomial distribution. These two algorithms make probabilistic assumptions or bounds on the joint distribution of the votes and the labels, but in our experiments, perform equally as well or slightly worse than CrowdSense.

Our work appears superficially similar to what is accomplished by polling processes. However, polling presumes knowledge of individuals' demographic characteristics, whereas our work does not consider the demographics of the labelers. Incorporating knowledge of labeler demographics, in order to predict the qualities of labelers, is our next direction for future work. Specifically, we plan to model the quality estimates of the labelers at a more fine-grained level, by considering a "context" for each labeler, as in contextual bandit problems. This will help in problems where certain labelers' expertise varies over topics or contexts. We hope to build models that better capture the correlations between the context and the votes, and quantify the uncertainty in our estimates of voting accuracy.

References

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proc. of the 23^{rd} annual ACM symposium on User interface software and technology (UIST)*, pages 313–322, 2010.

Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in two seconds: enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST)*, pages 33–42, 2011.

Chris Callison-Burch and Mark Dredze. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, 2010.

Wisam Dakka and Panagiotis G. Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 466–475, 2008.

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

Ofer Dekel and Ohad Shamir. Good learners for evil teachers. In *Proc. of the 26^{th} Annual International Conference on Machine Learning (ICML)*, 2009a.

Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *Proc. of the 22^{nd} Annual Conference on Learning Theory*, 2009b.

Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *The 23^{rd} Conference on Learning Theory (COLT)*, pages 346–358, 2010.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proc. of the 15th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 259–268, 2009.

Julie S. Downs, Mandy B. Holbrook, Steve Sheng, and Lorrie Faith Cranor. Are your participants gaming the system?: screening mechanical turk workers. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2399–2402, 2010.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

Dan Gillick and Yang Liu. Non-expert evaluation of summarization systems is risky. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 148–151. Association for Computational Linguistics, 2010.

Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. 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*, HLT '09, pages 27–35, 2009.

Michael Kaisser and John Lowe. Creating a research collection of question answer sentence pairs with amazons mechanical turk. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), 2008.

Gjergji Kasneci, Jurgen Van Gael, David Stern, and Thore Graepel. Cobayes: bayesian knowledge corroboration with assessors of unknown areas of expertise. In *Proceedings of the 4^{th} ACM International Conference on Web Search and Data Mining (WSDM)*, pages 465–474, 2011.

Edith Law and Luis von Ahn. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

M. Marge, S. Banerjee, and A.I. Rudnicky. Using the amazon mechanical turk for transcription of spoken language. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5270 –5273, march 2010.

Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 77–85, 2009.

Preslav Nakov. Noun compound interpretation using paraphrasing verbs: Feasibility study. In *Proceedings of the 13th international conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '08, pages 103–117, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85775-4.

Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, MIR '10, pages 557–566, 2010.

Susumu Ogawa and Frank Piller. Reducing the risks of new product development. *MITSloan Management Review*, 47(2), 2006.

Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI)*, pages 1403–1412, 2011.

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

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the* 14^{th} *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 614–622, 2008.

Padhraic Smyth, Michael C. Burl, Usama M. Fayyad, and Pietro Perona. Knowledge discovery in large image databases: Dealing with uncertainties in ground truth. In *KDD Workshop*, pages 109–120, 1994a.

Padhraic Smyth, Usama M. Fayyad, Michael C. Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1085–1092, 1994b.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, 2008.

Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. *Computer Vision and Pattern Recognition Workshop*, 0:1–8, 2008.

Elisabeth A. Sullivan. A group effort: More companies are turning to the wisdom of the crowd to find ways to innovate. *Marketing News*, 2010.

Byron C. Wallace, Kevin Small, Carla E. Brodley, and Thomas A. Trikalinos. Who should label what? instance allocation in multiple expert active learning. In *Proc. of the SIAM International Conference on Data Mining (SDM)*, 2011.

Simon K Warfield, Kelly H Zou, and William M Wells. Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. *IEEE Transactions on Medical Imaging (TMI)*, 23(7):903–21, 2004.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Jacob Whitehill, Paul Ruvolo, Ting fan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2035–2043, 2009.

Yan Yan, Romer Rosales, Glenn Fung, and Jennifer Dy. Modeling multiple annotator expertise in the semi-supervised learning scenario. In *Proceedings of the Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 674–682, Corvallis, Oregon, 2010a. AUAI Press.

Yan Yan, Rómer Rosales, Glenn Fung, Mark W. Schmidt, Gerardo Hermosillo Valadez, Luca Bogoni, Linda Moy, and Jennifer G. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. *Journal of Machine Learning Research - Proceedings Track (JMLR)*, 9:932–939, 2010b.

Yaling Zheng, S. Scott, and Kun Deng. Active learning from multiple noisy labelers with varied costs. In *10th IEEE International Conference onData Mining (ICDM)*, pages 639 –648, dec. 2010.

11. Appendix A

Proposition 1: *The score in (8) is always nonnegative.*

N_T = Total number of voters, which is an odd number so a majority always exists.
N_{voted} = number of voters that have voted already.
$N_{\text{current majority}}$ = number of voters that constitute the current majority vote. That is,

$$N_{\text{current majority}} > \frac{N_{\text{voted}}}{2} \tag{9}$$

N_{unvoted} = N$_T$ - N$_{\text{voted}}$ = number of labelers that have not yet voted.

To establish that the current majority vote is the true majority vote, we need

$$N_{\text{needed}} = \left\lceil \frac{N_T}{2} \right\rceil - N_{\text{current majority}}.$$

Using (9), $N_{\text{needed}} < \dfrac{N_T + 1}{2} - \dfrac{N_{\text{voted}}}{2} = \dfrac{N_{\text{unvoted}} + 1}{2}$

$$P_{\text{Bin}(N_{\text{unvoted}}, 0.5)}(X \geq N_{\text{needed}}) \geq P\left(X \geq \frac{N_{\text{unvoted}} + 1}{2}\right) \geq \frac{1}{2}.$$

To get the last inequality:

- If N_{unvoted} is odd, this probability is 1/2 with equality from the symmetry of the binomial distribution $\text{Bin}(N_{\text{unvoted}}, \frac{1}{2})$.

- If N_{unvoted} is even, this probability is greater than 1/2, since there is probability mass at $X = \dfrac{N_{\text{unvoted}} + 1}{2}$.

12. Appendix B

In Section 9, we described a flipping heuristic for CrowdSense.Ind where we flip the votes of labelers with quality estimates below 0.5. In Algorithm 4, we present the pseudocode of CrowdSense.Ind with this heuristic. Figure 10 presents the tradeoff curves for CrowdSense.Ind with and without flipping the votes of the labelers with quality estimates less than 0.5, averaged over 100 runs. A remarkable difference between these two curves are observed only in experiments with ChemIR dataset. In experiments with all other datasets, both methods give similar results.

Algorithm 4 Pseudocode for CrowdSense.Ind with flipping.

1. **Input:** Examples $\{x_1, x_2, \ldots, x_N\}$, Labelers $\{l_1, l_2, \ldots, l_M\}$, confidence threshold ε, smoothing parameter K.

2. **Define:** $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates. P_+ is the prior probability of a +1 majority vote.

3. **Initialize:** $a_i \leftarrow 0,\ c_i \leftarrow 0$ for $i = 1, \ldots, M$.

4. **Loop for** $t = 1, ..., N$

   (a) $Q_{it} = \frac{a_i+K}{c_i+2K}, \quad \forall i$

   (b) $\forall i$, **If** $Q_{it} < \frac{1}{2}$ **then** $Q_{it} \leftarrow 1 - Q_{it}, \quad F_i = 1$ **else** $F_i = 0$

   (c) Update L_Q with the new quality estimates.

   (d) Select 3 labelers and get their votes. $S_t = \{l^{(1)}, l^{(2)}, l^{(k)}\}$, where k is chosen uniformly at random from the set $\{3, \ldots, M\}$.

   (e) **if** $F_i = 1, V_{it} \leftarrow -V_{it}$ (flip labeler's vote)

   (f) $V_{it}^{\text{bin}} = \frac{(V_{it}+1)}{2}, \quad \forall i \in S_t$

   (g) $\psi_{it} = Q_{it}^{V_{it}^{\text{bin}}} (1 - Q_{it})^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$

   (h) $\theta_{it} = (1 - Q_{it})^{V_{it}^{\text{bin}}} Q_{it}^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$

   (i) **Loop for** candidate $= 3 \ldots M,$ candidate $\neq k$

      i. $$f(x_t|\text{votes}) = \frac{1}{1 + \frac{(\prod_{i \in S_t} \theta_{it})(1 - P_+)}{(\prod_{i \in S_t} \psi_{it}) P_+}}$$

      ii. $\psi_{\text{candidate}} = 1 - Q_{\text{candidate},t}$

      iii. $\theta_{\text{candidate}} = Q_{\text{candidate},t}$

      iv. $$f(x_t|\text{votes}, V_{\text{candidate},t}) = \frac{1}{1 + \frac{(\prod_{i \in S_t} \theta_{it}) \theta_{\text{candidate}} (1 - P_+)}{(\prod_{i \in S_t} \psi_{it}) \psi_{\text{candidate}} P_+}}$$

      v. **If** $f(x_t|\text{votes}) \geq 0.5$ **and** $f(x_t|\text{votes}, V_{\text{candidate},t}) < 0.5 + \varepsilon$
      ShouldBranch = 1

      vi. **If** $f(x_t|\text{votes}) < 0.5$ **and** $f(x_t|\text{votes}, V_{\text{candidate},t}) > 0.5 - \varepsilon$
      ShouldBranch = 1

      vii. **If** ShouldBranch = 1 **then** $S_t = \{S_t \cup \text{candidate}\}$, get the candidate's vote
      else Don't need more labelers, break out of loop.

   (j) $\hat{y}_t = 2 \times \mathbb{1}_{[f(x_t|\text{votes})>0.5]} - 1$

   (k) $\forall i \in S_t$ where $V_{it} = \hat{y}_t, \quad a_i = a_i + (1 - F_i)$

   (l) $\forall i \in S_t$ where $V_{it} \neq \hat{y}_t, \quad a_i = a_i + F_i$

   (m) $c_i = c_i + 1, \quad \forall i \in S_t$

5. **End**

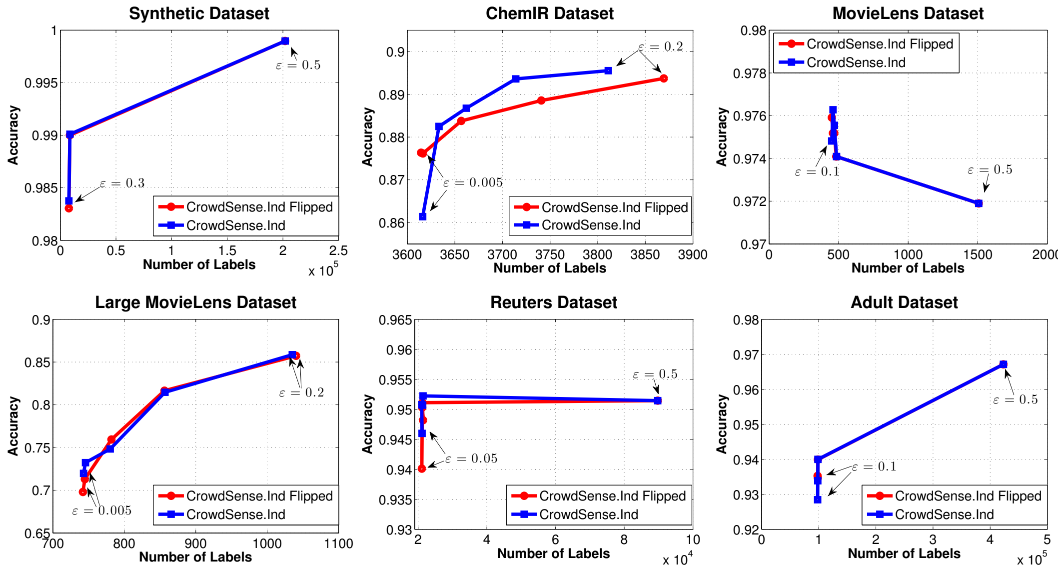


Figure 10: Tradeoff curves for CrowdSense.Ind with and without the flipping heuristic.