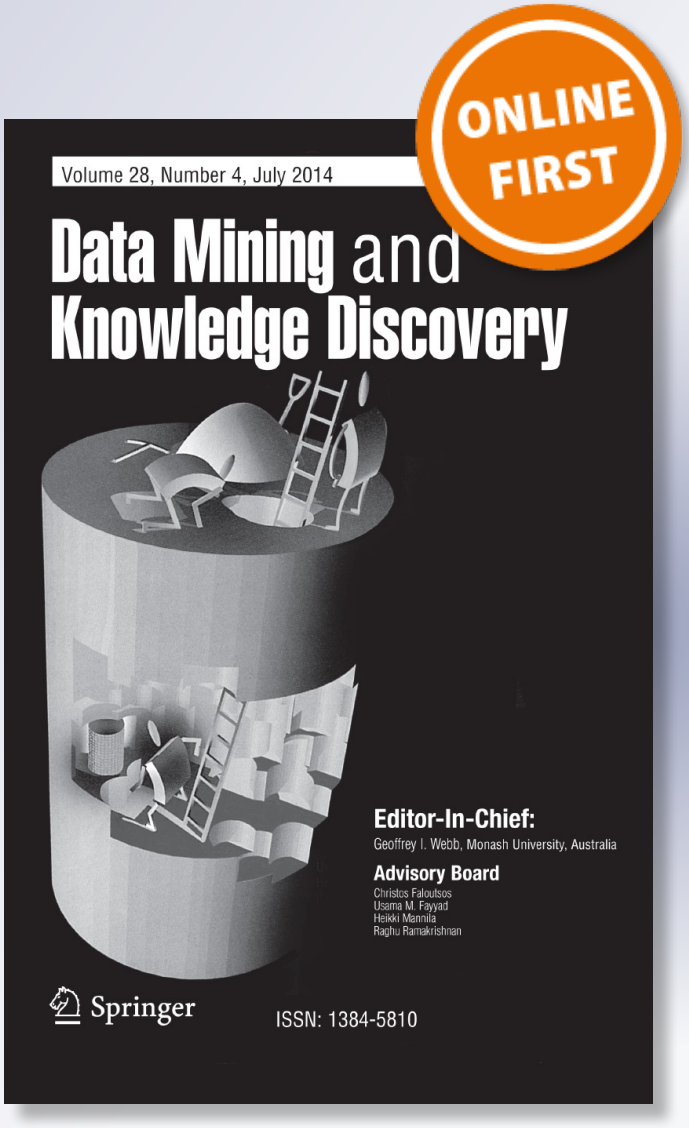
Approximating the crowd

Şeyda Ertekin, Cynthia Rudin & Haym Hirsh

Data Mining and Knowledge Discovery

ISSN 1384-5810

Data Min Knowl Disc
DOI 10.1007/s10618-014-0354-1



Springer

Data Min Knowl Disc
DOI 10.1007/s10618-014-0354-1

Approximating the crowd

Şeyda Ertekin · Cynthia Rudin · Haym Hirsh

Received: 17 February 2013 / Accepted: 13 May 2014

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 items come one at a time. CrowdSense dynamically samples subsets of the crowd based on an exploration/exploitation criterion. The algorithm produces a weighted combination of the subset's votes that approximates the crowd's opinion. We then introduce two variations of CrowdSense that make various distributional approximations to handle distinct crowd characteristics. In particular, the first algorithm makes a statistical independence approximation of the labelers for large crowds, whereas the second algorithm finds a lower bound on how often the current subcrowd agrees with the crowd's 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.

Responsible editor: Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, Filip Zelezny.

Electronic supplementary material The online version of this article (doi:10.1007/s10618-014-0354-1) contains supplementary material, which is available to authorized users.

Ş. Ertekin (✉) · C. Rudin
MIT CSAIL, Sloan School of Management, and Center for Collective Intelligence,
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: seyda@mit.edu

C. Rudin
e-mail: rudin@mit.edu

H. Hirsh
Department of Computer Science and Information Science, Cornell University, Ithaca, NY, USA
e-mail: hirsh@cs.cornell.edu

Published online: 14 June 2014

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

1 Introduction

Our goal is to determine the majority opinion of a crowd on a series of questions ("examples"), where each member's opinion is obtained at a cost, and our total budget is limited. 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 approves 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. In crowdsourcing problems, either there is no ground truth (as the ultimate goal is to determine a judgment), or there is ground truth, but with no possibility of its being revealed. The application domain of *answer services* is relevant to the problem of approximating the crowd. Here are some specific example applications:

- VizWiz[1] (Bigham et al. 2010) is an iPhone app used by visually impaired people to obtain answers to questions about their surroundings. VizWiz queries multiple Amazon Mechanical Turkers, and on its website, VizWiz issues the statement "VizWiz will recruit multiple answers from different web workers to help you better gauge the reliability of answers retrieved in this way." There may or may not be ground truth in the questions posed on VizWiz, but either way, the majority vote of the crowd is likely to be the desired answer to the query. The more turkers you query, the more confident you can be about the answer. However, if several turkers need to be queried, it will take longer (with a cost) for the visually impaired person to retrieve the answer. If a system existed that could learn over time, as queries proceed, who are the reliable members of the crowd that agree with its majority vote, then the system could potentially be a valuable contributor to services like VizWiz.
- IQ Engines[2] is a crowdsourced image recognition platform. It uses a combination of computer vision software and human labelers to identify objects in photographs. Over time, as it is queried more and more, it would be useful for the system to learn who are the most reliable humans to answer the queries. This way, fewer humans need to be involved over time, and only the most reliable humans need to be hired.

[1] http://vizwiz.org

[2] http://iqengines.com/

Again, there may or may not be ground truth, but the majority vote of the crowd is likely to be a desirable, or at least acceptable answer, by computer vision standards.

– Polling is by definition crowdsourced. 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. These companies would like, with minimal effort, cost, and speed, to estimate the crowd's majority vote accurately. 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 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 as in the *answer services* applications above, 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 many 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 fewer 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 three parts: In the first part (Sect. 3), we introduce the problem of approximating the crowd, and the notion of the frontier of cost and accuracy. We solve analytically for the expected values of the extreme points on the frontier.

In the second part of the paper (Sect. 4), 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 labelers, and exploration so that enough data are obtained about each labeler to ensure that we learn each labeler's accuracy sufficiently well. We present experiments across several datasets and in comparison with several baselines in Sect. 6, where the cost constraint parameter was set over several different values, and variants were computed over 100 runs to ensure the quality of the solution. We then discuss the effects of CrowdSense's parameters in Sect. 7, and the effects of the various subcomponents in Sect. 8.

The third part of the paper presents probabilistic variations of CrowdSense, in Sect. 9. 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 with respect to the majority vote. In Sects. 9.1 and 9.2 of the paper, we introduce two variations of CrowdSense, called CrowdSense.Ind and CrowdSense.Bin, that make specific distributional approximations to handle distinct crowd characteristics. In particular, the first algorithm makes a statistical independence approximation of the probabilities for large crowds, whereas the second algorithm finds a lower bound on how often the current subcrowd agrees with the crowd majority vote, using the binomial distribution. For both CrowdSense.Ind and CrowdSense.Bin, even though explicit probabilistic approximations 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. Experiments comparing the three CrowdSense variants are in Sect. 10.

In the supplementary file[3], in Section A, we present a proposition for CrowdSense.Bin that states that the computed scores are non-negative. Section B presents a "flipping" technique – a heuristic for taking the opposite vote of labelers that have low quality estimates. In Section C, we discuss the possibility of learning the labelers' quality estimates using machine learning. Section D presents the runtime performance of the algorithms. Section E presents the effect of gold standard data in initialization.

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.

2 Related work

The low cost of crowdsourcing labor has increasingly led to the use of resources such as Amazon Mechanical Turk[4] (AMT) to label data for machine learning purposes, where

[3] http://github.com/CrowdSense/SupplementaryMaterial

[4] http://www.mturk.com

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 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 Ipeirotis et al. 2010; 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 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 (it is precisely the majority vote) 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 Sect. 5 as one of the baselines to which we compare our results. 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 smoothed estimates of the labelers' qualities. Early results of this work appeared in Collective Intelligence 2012 conference (Ertekin et al. 2012).

3 Fundamentals of approximating the crowd

We define an "approximating the crowd" problem to be characterized by a sequence of random vectors $\mathbf{V}_t$ for $t = 1 \ldots T$, where each $\mathbf{V}_t$ is drawn iid from an unknown distribution μ on $\{-1, 1\}^M$, i.e. $\mathbf{V}_t \sim \mu(\{-1, 1\}^M)$. $\mathbf{V}_t$ represents the set of votes that are given by all M labelers at time t, if they were selected to vote. The value Y_t for each t is computed deterministically as a function of random variable $\mathbf{V}_t$ by

$$Y_t = \begin{cases} 1 & \text{if } \sum_{i=1}^{M} V_{t_i} > 0 \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

An algorithm for "approximating the crowd" is a policy π that, at each time t, determines:

- Which votes V_{t_i} should be revealed, where each vote is obtained at a given fixed unit cost, and in which order the votes should be obtained,
- When to stop requesting votes (when we are certain enough to estimate Y_t),
- How to combine votes to obtain an estimate of Y_t, called $\hat{Y}_t$.
- The total cost is $\text{Cost}(\pi) = \#\text{Votes}(\pi)$.

The accuracy of algorithm π is defined as $\text{Reward}(\pi) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}_{\left[\hat{Y}_t = Y_t\right]}$. The goal of an algorithm π for approximating the crowd is to maximize reward subject to a constraint on the cost, given policy π for approximating the crowd. These constraints can be either hard or soft. Thus the goal of approximating the crowd can be written in either of the two ways below:
Hard-constrained

$$\max_{\pi} \mathbb{E}_{\{V_t\}_t : V_t \sim \mu} \left[\text{Reward} \mid \pi\right] \quad \text{s.t.} \quad \text{Cost}(\pi) \leq C_{\text{hard}}.$$

Soft-constrained

$$\max_{\pi} \mathbb{E}_{\{V_t\}_t : V_t \sim \mu} \left[\text{Reward} - C_{\text{soft}} \cdot \text{Cost}(\pi) \mid \pi\right].$$

The user defines which goal (hard or soft) is appropriate for the problem.

Algorithms for approximating the crowd must use their budget to balance exploring the qualities of the labelers and exploiting the high quality labelers.When comparing two algorithms against each other, one can consider reward and cost along separate dimensions. It is possible that π_1 could be better than π_2 in both reward and cost, in which case π_1 dominates π_2. That is, We say that algorithm π_1 *dominates* π_2 on a particular dataset if:

$$\begin{aligned} &\text{Cost}(\pi_1) < \text{Cost}(\pi_2) \text{ and } \text{Reward}(\pi_1) \geq \text{Reward}(\pi_2), \text{ or} \\ &\text{Cost}(\pi_1) \leq \text{Cost}(\pi_2) \text{ and } \text{Reward}(\pi_1) > \text{Reward}(\pi_2). \end{aligned}$$

In the second condition above, π_1 is able to achieve a higher accuracy with a lower cost than π_2, meaning it achieves a better value of the hard-constrained objective function (higher reward for a fixed cost). If either of the two conditions above for domination are met, π_1 achieves a better value of the soft-constrained objective than π_2.

As we adjust C_{hard} for the hard-constrained problem, we can trace out an *efficient frontier* of solutions; these are the solutions that maximize expected accuracy for each fixed cost. The algorithms we present in this paper each have a "cost" parameter that the user can adjust. By adjusting the parameter, we obtain different accuracy values for each possible cost. This allows us to empirically trace out a *frontier* of solutions for the algorithm. It is difficult to determine the efficient frontier, as an optimal policy would depend on the distribution of the labelers.

3.1 Understanding the frontier

Parts of the frontier can be analytically determined under specific conditions, for instance when $\mu(\{-1, 1\}^M)$ is a product distribution, where each labeler chooses their label in a way that is conditionally independent from other labelers. We construct such a distribution as follows. To ensure that there is no prior information available to the algorithm, we start with a binary signal chosen uniformly at random, taking values +1 or -1, that is: $X_t^{\text{signal}} \sim [\text{Bernoulli}(0.5)] \times 2 - 1$. Each labeler agrees with X_t^{signal} with

probability p_{all}. Assume p_{all} is known, $p_{\text{all}} > 0.5$ (without loss of generality, since we can reverse the vote if $p_{\text{all}} < 0.5$). As usual

$$Y_t = \begin{cases} 1 & \text{if } \sum_{i=1}^{M} V_{t_i} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

We can solve analytically for the two important points on the frontier: the expected accuracy for the nontrivial policy that makes one vote per example, Cost=T $\cdot u$, where u is the unit cost of one vote from one labeler (Theorem 1), and the expected cost for the optimal policy achieving perfect accuracy, Reward=1 (Theorem 2). We present these values in the case of M total labelers.

Theorem 1 *Assume M labelers, with the product distribution described above. For the (low-cost) policy that pays for exactly one vote per example at each time t, the expected accuracy is*

$$\mathbb{E}_{\{\mathbf{V}_1,\ldots,\mathbf{V}_T \sim \mu(\{-1,1\}^M)\}} = p_{all} \cdot \left[\sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, p_{all}) \right] + (1 - p_{all}) \cdot \left[\sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, 1-p_{all}) \right],$$

where $\text{Bin}(x, n, p)$ *represents the x'th entry in the binomial distribution with parameters n and p, i.e.* $\text{Bin}(x, n, p) = \binom{n}{x} p^x (1-p)^{n-x}$.

Here is the result for the other extreme point.

Theorem 2 *The expected cost of the optimal policy that achieves perfect accuracy in predicting the crowd's majority vote is*

$$\Omega = T \sum_{j=\binom{M+1}{2}}^{M} \binom{j-1}{\frac{M-1}{2}} \left[p_{all}^{\frac{M+1}{2}} (1-p_{all})^{j-\frac{M+1}{2}} + (1-p_{all})^{\frac{M+1}{2}} p_{all}^{j-\frac{M+1}{2}} \right] \times j.$$

This policy purchases as many labels as necessary to determine the crowd's majority vote with probability 1.

Note that the extreme point in Theorem 2 comes from the optimal policy for achieving maximal reward, and is thus on the efficient frontier. We lead up to the proofs of Theorem 1 and Theorem 2.

Since each p_{all} is known, we do not learn labeler qualities over time. Thus, the expected value of the accuracy over time is the expected value of accuracy for a single measurement. Thus, we eliminate the "t" index in the proofs. We define V_i to be the vote from labeler i.

Proof (Theorem 1) Our algorithm chooses $\hat{Y} = V_M$ for all t, that is, we will pay for only one labeler's vote per time, namely the M^{th} labeler's vote, without loss of generality. Without obtaining at least one labeler's vote, the accuracy would be exactly 0.5. Note that p_{all} is *not* the accuracy of the labeler with respect to the crowd's majority, it is simply the probability of agreement with the signal. We need to calculate the accuracy of $\hat{Y} = V_M$, which is the probability that labeler M agrees with the majority, $\frac{1}{2}P(Y = 1|V_M = 1) + \frac{1}{2}P(Y = -1|V_M = -1)$. Due to the symmetry of the problem, we have $P(Y = 1|V_M = 1) = P(Y = -1|V_M = -1)$. Thus, it is sufficient to calculate $P(Y = 1|V_M = 1)$, which is the probability that the majority vote is 1 given that labeler votes 1.

$$\begin{aligned} P(Y = 1|V_M = 1) &= P(Y = 1|V_M = 1, X^{\text{signal}} = 1)P(X^{\text{signal}} = 1|V_M = 1) \\ &+ P(Y = 1|V_M = 1, X^{\text{signal}} = -1)P(X^{\text{signal}} = -1|V_M = 1) \end{aligned} \tag{2}$$

where

$$\begin{aligned} P(X^{\text{signal}} = 1|V_M = 1) &= \frac{P(V_M = 1|X^{\text{signal}} = 1)P(X^{\text{signal}} = 1)}{P(V_M = 1)} \\ &= P(V_M = 1|X^{\text{signal}} = 1) = p_{\text{all}} \end{aligned}$$

since $P(X^{\text{signal}} = 1) = \frac{1}{2}$ and $P(V_M = 1) = \frac{1}{2}$. Also,

$$P(X^{\text{signal}} = -1|V_M = 1) = 1 - P(V_M = 1|X^{\text{signal}} = 1) = 1 - p_{\text{all}}.$$

To calculate the other terms of (2) we need to compute

$$\begin{aligned} &P(Y = 1|V_M = 1, X^{\text{signal}} = 1) \\ &\quad = P\left(\text{at least } \frac{M-1}{2} \text{ of voters } 1, \ldots, \text{ M-1 vote } 1|X^{\text{signal}} = 1\right) \\ &\quad = \sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, p_{\text{all}}), \end{aligned}$$

since we are summing independent Bernoulli random variables. Further,

$$\begin{aligned} &P(Y = 1|V_M = 1, X^{\text{signal}} = -1) \\ &\quad = P\left(\text{at least } \frac{M-1}{2} \text{ of voters } 1, \ldots, \text{M-1 vote } 1|X^{\text{signal}} = -1\right) \\ &\quad = \sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, 1-p_{\text{all}}). \end{aligned}$$

Putting it together,

$$P(Y=1|V_M=1) = \left[\sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, p_{\text{all}})\right] p_{\text{all}} + \left[\sum_{x=\frac{M-1}{2}}^{M-1} \text{Bin}(x, M-1, 1-p_{\text{all}})\right] (1-p_{\text{all}}).$$

□

Proof (Theorem 2) Let us say that the (hidden) majority class label is positive without loss of generality. We will purchase labels until we have gathered $\frac{M+1}{2}$ positive labels; if we have at least that many positive labels, we will have determined that the majority class must be positive.

When we stop gathering labels, the final label we will have collected is the $\frac{M+1}{2}$'st positive label. Its probability of being positive is p_{all} if the signal is positive or $1-p_{\text{all}}$ if the signal is negative.

Prior to paying for the final positive label, we must have purchased exactly $\frac{M+1}{2} - 1 = \frac{M-1}{2}$ other positive labels and at most M total labels. Thus, we sum over the possibilities of purchasing j labels, where j ranges from $\frac{M+1}{2}$ total labels (this is the case where we had been lucky to purchase all the positive labels we need in a row) to $j = M$ total labels (where we had purchased $\frac{M-1}{2}$ positive labels and $\frac{M-1}{2}$ negative labels and the last vote is the deciding vote).

Let us fix j, where we say we pay for j total labels, where the final label is positive, and there were $\frac{M-1}{2}$ other positive labels purchased among the remaining $j-1$ labels purchased. The probability of this happening is

$$p_{\text{all}}^{\frac{M+1}{2}} (1-p_{\text{all}})^{j-\frac{M+1}{2}}$$

when the signal is positive, and

$$(1-p_{\text{all}})^{\frac{M+1}{2}} p_{\text{all}}^{j-\frac{M+1}{2}}$$

when the signal is negative. The number of ways this could have occurred is $\binom{j-1}{\frac{M-1}{2}}$. Thus the expected number of labels we need to pay for over T times is

$$\Omega = T \sum_{\left(j=\frac{M+1}{2}\right)}^{M} \binom{j-1}{\frac{M-1}{2}} \left[p_{\text{all}}^{\frac{M+1}{2}} (1-p_{\text{all}})^{j-\frac{M+1}{2}} + (1-p_{\text{all}})^{\frac{M+1}{2}} p_{\text{all}}^{j-\frac{M+1}{2}}\right] \times j.$$

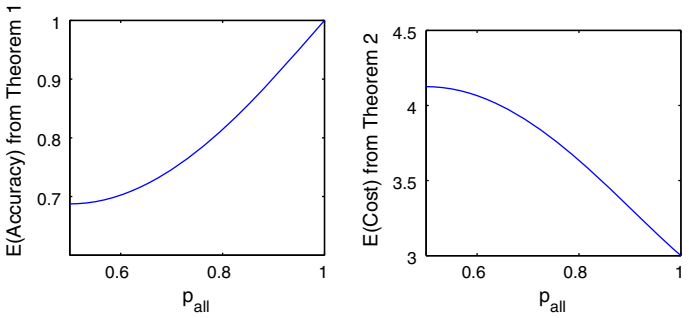


Fig. 1 Results from Theorems 1 and 2

The two theorems above pinpoint two ends of the frontier. The strategy used in Theorem 1 has a fixed cost of T total votes over the T examples, and attains low accuracy, provided within the theorem. The strategy used in Theorem 2 has perfect accuracy, but has a high cost, provided within the theorem. In general we would like to be somewhere along (but not necessarily at either end of) the frontier. CrowdSense and its variants have internal parameters that govern where on its frontier the result will lie. In the experimental sections that follow, we plot the result of each algorithm in cost-accuracy space. If an algorithm achieves comparably better accuracy with the same cost along the full frontier, then it dominates its competitor for that dataset across all possible hard cost constraints.

A plot of the expected accuracy provided by Theorem 1 as a function of p_{all}, as well as a plot of the expected cost per time as a function of p_{all} from Theorem 2 are found in Fig. 1, for 101 labelers. Note in particular that even if only one labeler is chosen when $p_{\text{all}} = 0.5$, the expected accuracy from Theorem 1 is well above 0.5. This is because the labeler whose vote we paid for contributes to the crowd's majority vote, and thus contains some information about the true majority. The unit cost per labeler in Fig. 1 is $u = 1$.

We can connect the theoretical endpoints of the frontier with experimental results. Figure 2 shows the theoretical results from Theorems 1 and 2 on the frontier, for 7 labelers and 50 K examples, where p_{all} was set at 0.5. This figure also shows results from CrowdSense.Bin, using a dataset simulated with the same specifications.

Now that we have formally defined "approximating the crowd," we present CrowdSense for approximately solving it. It is difficult to create an algorithm whose solution lies near the efficient frontier for most distributions of labelers, but CrowdSense seems to be able to adapt nicely to a wide variety of distributions.

4 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

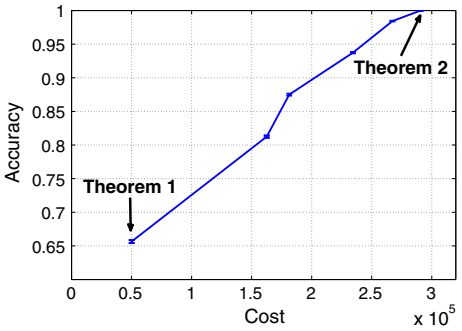


Fig. 2 We bridge theoretical and empirical results by plotting the frontier from CrowdSense.Bin, and including results from Theorems 1 and 2

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:

$$c_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{[i \in S_{\tilde{t}}]} \tag{3}$$

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

$$a_{it} := \sum_{\tilde{t}=1}^{t} \mathbb{1}_{\left[i \in S_{\tilde{t}}, V_{i\tilde{t}} = V_{S_{\tilde{t}}\tilde{t}}\right]} \tag{4}$$

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

$$Q_{it} := \frac{a_{it} + K}{c_{it} + 2K} \tag{5}$$

where t is the number of examples that we have collected labels for and K is a smoothing parameter. Q_{it} is a smoothed 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

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. **Initialize:** $a_{i1} \leftarrow 0$, $c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$ and $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$, $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$ random permutation of labeler id's
3. **For** $t = 1, ..., N$
   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K}$, $i = 1, \ldots, M$. Update $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates. If quality estimates are identical, randomly permute labelers with identical qualities to attain an order.
   (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. $\text{Score}(S_t) = \sum_{i \in S_t} V_{it} Q_{it}$, $l_{\text{candidate}} = l^{(j)}$.
      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) *Return* 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 $V_{it} = V_{S_t t}$, $a_{it} \leftarrow a_{it} + 1$
   (f) $\forall i \in S_t$, $c_{it} \leftarrow c_{it} + 1$
4. **End**

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:

$$\frac{|\text{Score}(S_t)| - Q_{l_{\text{candidate}},t}}{|S_t| + 1} < \varepsilon \tag{6}$$

where ε controls the level of uncertainty we are willing to permit, $0 < \varepsilon \leq 1$. If (6) 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.

5 Datasets and baselines

In order for us to evaluate the quality of a system for approximating the crowd, we require databases with many examples, for which *all* of the labelers' votes are known on each example (so the majority can be calculated). This allows an *unbiased, offline* evaluation to be performed. Ground truth labels may or may not be present; as in the *answer services* applications, even if ground truth exists, we assume there is no realistic way to obtain it. One of the most commonly known crowdsourced databases was used in the Netflix Contest[5]. This database contains 100 million anonymous movie ratings, and could definitely be used to predict the crowds opinion on movies, even though it was not explicitly designed for that purpose (however it is no longer publicly available). For our experiments, we used 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 common prediction of these models.

This paper thus develops the largest collection of complete crowdsourced databases available with the necessary characteristics for these types of experiments. We also varied the amount of various types of noise in the datasets, to account for a broader type of data than is publicly available, and to provide a type of sensitivity analysis. The number of examples in each dataset and the true accuracies of the labelers are shown in Table 1. Our datasets are publicly available[6].

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.

MovieLens is a movie recommendation dataset of user ratings on a collection of movies, and our goal is to find the majority vote of these reviewers. The dataset is originally very sparse, meaning that only a small subset of (human) 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 at a cost. We mapped the original rating scale of [0–5] to votes of -1, 1 by using 2.5 as the threshold. The resulting dataset has the benefits of having both human labelers and the completeness required to evaluate algorithms for approximating the crowd. There is no ground truth in movie ratings, other than the crowd's majority vote.

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;

[5] http://www.netflixprize.com

[6] Dataset are available at http://github.com/CrowdSense/Datasets

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

MovieLens	ChemIR	Reuters	Adult	SpamBase	MTurk
Number of examples					
137	1,165	6,904	32,561	2,300	673
Labeler accuracies					
48.17(L1)	50.72(L1)	80.76(L1)	81.22(L1)	86.30(L1)	52.60(L1)
89.78(L2)	46.78(L2)	83.00(L2)	80.59(L2)	86.35(L2)	63.30(L2)
93.43(L3)	84.46(L3)	89.70(L3)	86.22(L3)	91.22(L3)	54.98(L3)
48.90(L4)	88.41(L4)	82.98(L4)	87.63(L4)	94.04(L4)	79.64(L4)
59.12(L5)	86.69(L5)	88.12(L5)	91.12(L5)	75.91(L5)	61.52(L5)
96.35(L6)	87.46(L6)	87.04(L6)	94.11(L6)	82.04(L6)	72.22(L6)
87.59(L7)	49.52(L7)	95.42(L7)	56.68(L7)	68.39(L7)	74.00(L7)
54.01(L8)	78.62(L8)	80.21(L8)	85.51(L8)	90.83(L8)	
47.44(L9)	82.06(L9)	78.68(L9)	81.32(L9)	94.35(L9)	
94.16(L10)	50.12(L10)	95.06(L10)	85.54(L10)		
95.62(L11)	50.98(L11)	82.88(L11)	79.74(L11)		
		71.57(L12)	84.86(L12)		
		87.54(L13)	96.71(L13)		

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. The ground truth in the ChemIR dataset is debatable: it is possible that the crowd has identified citations that *should* have been made in the patent, but were not. So it is possible that the crowd's majority vote could be more valuable to future patent citation suggestion than an algorithm that actually models ground truth citations.

Reuters is a popular dataset of articles that appeared on 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 algorithms for approximating the crowd. 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 persons annual income exceeds $50 K/yr.

Spambase is another popular dataset from the UCI repository comprised of content features extracted from spam and non-spam email messages and the task is to distinguish these two types. Similar to the Reuters and Adult datasets, we trained various

machine learning algorithms with various settings on half of the dataset, and used their predictions on the other half as the votes for those examples. In total, our dataset contains 2,300 examples, each voted on by 9 labelers.

Finally, we experimented with the HITSpam dataset[7], which includes descriptions of tasks posted on the Amazon Mechanical Turk marketplace. This dataset was compiled by asking workers on Mechanical Turk to examine whether a task is "legitimate" or if the purpose of the task is to game social media metrics, such as asking workers to follow a user on Twitter, to "like" a video on YouTube, etc. This dataset contains 5,840 tasks voted on by 135 labelers for a total of 28,354 votes. While the dataset is large in terms of the total number of votes, there are very few tasks that have been voted on by the same set of labelers. Even if we consider the 5 labelers that have labeled the most tasks, there are only 89 tasks that are voted on by all of them. If we consider the top 7 labelers, there are no tasks that are voted on by all of them. In order to obtain a sizable set of tasks that have votes from all labelers, we proceeded as follows: First, we identified pairs of labelers l_i and l_j where both labelers have labeled at least 10 tasks in common, and the votes of l_i and l_j on all common-labeled tasks are identical. We considered such a pair as a single labeler, and the votes of this labeler were set to the union of the labels of l_i and l_j. From there, we selected the 7 labelers that labeled the most tasks, and we selected the tasks that have been labeled by at least 5 of the labelers. In total, there were 673 tasks that matched this criteria. For the cases in which a labeler had not labeled a task, we generated a label randomly according to the labeler's accuracy (i.e. agreement with the majority vote) on the tasks that she has labeled.

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 performance.

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. In Section C of the supplementary file, we also discuss the possibility of learning the labelers' quality estimates using machine learning.

Another baseline that we use to compare CrowdSense is 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:

[7] http://github.com/ipeirotis/Get-Another-Label/tree/master/data/HITspam-UsingMTurk

$$UI(a) = m(a) + t_{\frac{\alpha}{2}}^{(n-1)} \frac{s(a)}{\sqrt{n}} \tag{7}$$

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. In Section D of the supplementary file, we present the runtime performance of CrowdSense and IEThresh and demonstrate that CrowdSense achieves higher accuracy even though it collects fewer labels than IEThresh.

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 by 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 subcrowd, similar to CrowdSense.

We compared CrowdSense against two additional methods, namely labeling quality uncertainty (LU) (Sheng et al. 2008) and new label uncertainty (NLU) (Ipeirotis et al. 2013), that provide smoothed estimates of the uncertainty of the labels for each example. These algorithms work offline, in that they are allowed to go back to any selected example they choose and request another label. (This could potentially give these algorithms a large advantage over CrowdSense and IEThresh.) In LU, given l_p positive and l_n negative labels for an example, the posterior probability of the true label $p(y)$ follows a Beta distribution $B(l_p + 1, l_n + 1)$ and the level of uncertainty is measured as the tail probability below the labeling decision threshold of 0.5. The example that is closest to this threshold is the one that has the most uncertain label and the next set of labels should be selected for this instance. This calculation makes the approximation that all labelers have the same quality when labeling a given example, which is not an assumption of CrowdSense.

For NLU, any example that we have observed $l_p > l_n$ labels has the posterior probability for the positive class

$$Pr(+|l_p, l_n) = \left(1 + \frac{1 - Pr(+)}{Pr(+)} \cdot \frac{\left((l_n + l_p)!\right)^2}{(2l_n)! \cdot (2l_p)!}\right)^{-1}$$

where $Pr(+)$ denotes the prior for the positive class that can be computed iteratively using a marginal maximum likelihood algorithm. For examples with $l_n > l_p$ labels, the posterior above is symmetric. Similar to the LU case, the example with the most uncertain label is the one that has the score closest to 0.5 and the algorithm queries for new labels for that example in the next iteration. We used the setting of LU and NLU as they are presented in (Sheng et al. 2008) and (Ipeirotis et al. 2013) respectively.

6 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 3 shows the comparison of CrowdSense against baseline (b) and IEThresh. On the subplots in Fig. 3, 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, 84.84 and 95.25 %, for Adult 83.94 and 95.03 %, for Mechanical Turk 65.46 and 77.12 %, and for Spambase the values are 85.49 and 93.43 %. The results indicate that uniformly across different values of ε, CrowdSense consistently achieved the highest accuracy against these baselines, indicating that CrowdSense uses any fixed budget more effectively than the baselines. 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. Also, the results demonstrate that asking for labels from labelers at random may yield poor performance for representing 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.

For the LU and NLU experiments, we initialized both algorithms by randomly selecting three labelers and acquiring their votes on each example in the dataset. At each round of collecting more labels for the most uncertain example, we acquired two additional labels to prevent ties. We did not specify a stopping criteria for either LU or NLU, so both algorithms ran until acquiring all labels for all examples in the dataset (therefore both algorithms eventually converge to 100 % accuracy). The comparison of CrowdSense with LU and NLU is presented in Fig. 4. The results demonstrate that CrowdSense achieves better general performance, where its impact is most pronounced on the MovieLens, ChemIR, Mechanical Turk and Spambase datasets. We remark that

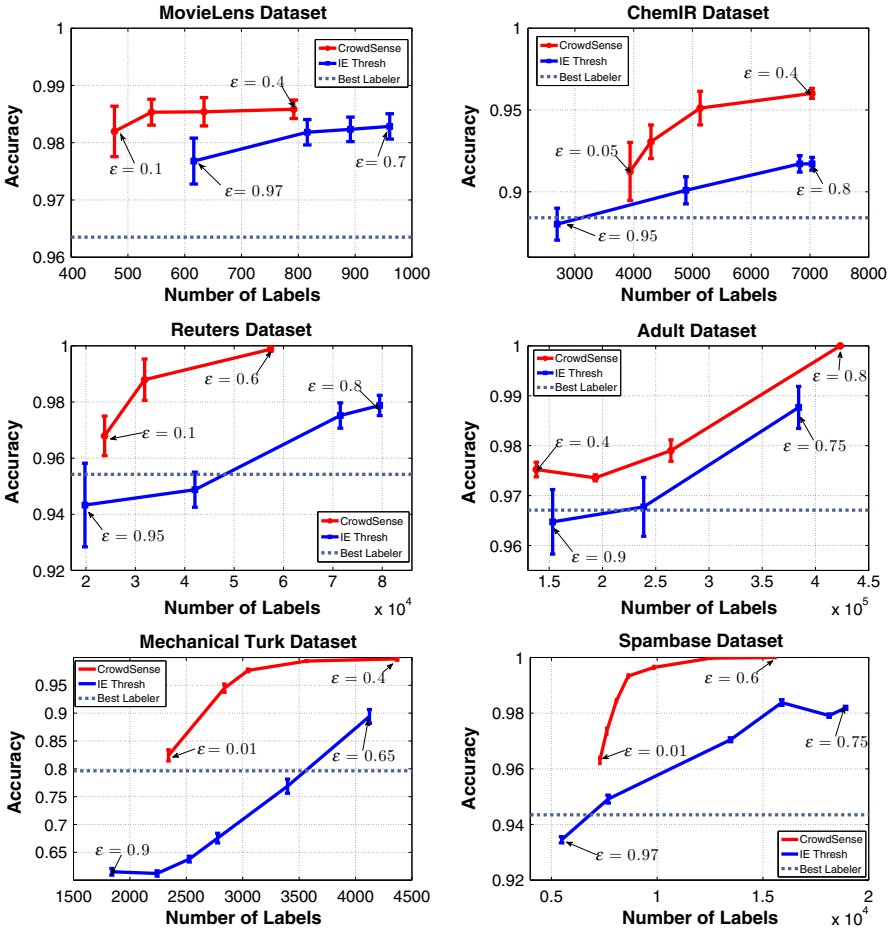


Fig. 3 Tradeoff curves for CrowdSense, baseline (b), and IEThresh, averaged over 100 runs. The x-axis is the total number of votes (the total cost) used by the algorithm to label the entire dataset. The y-axis indicates the accuracy on the full dataset

LU and NLU are permitted to look backwards in time, where CrowdSense is not. LU and NLU make very strong explicit probabilistic approximations (i.e. independence) that are not necessarily true. Later in the paper we will present CrowdSense.Bin and CrowdSense.Ind that also make explicit probabilistic approximations that are not true, and they also do not yield better performance than CrowdSense. CrowdSense's probabilistic assumptions are more implicit, and translate into the smoothed quality estimates of the labelers and the use of the regime of uncertainty.

7 Effect of parameters

In this section, we discuss the impact of various settings of ε and K on CrowdSense's performance. Evaluation of our specific choices that we made for exploration and

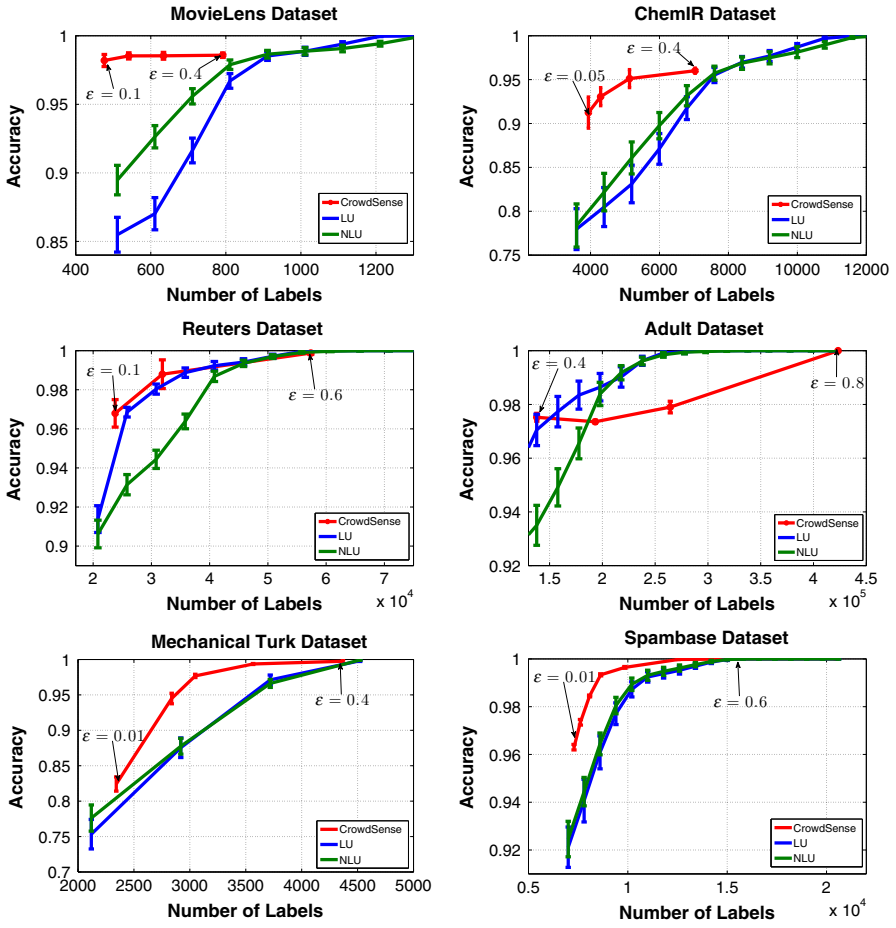


Fig. 4 Tradeoff curves for CrowdSense, LU and NLU, averaged over 100 runs. The x-axis is the total number of votes (the total cost) used by the algorithm to label the entire dataset. The y-axis indicates the accuracy on the full dataset

exploitation in the different components of CrowdSense is presented in Sect. 8. We also present experimental results that demonstrate the effect of initialization with gold standard data in Section E of the supplementary file.

7.1 Effect of the ε parameter

As discussed throughout Sects. 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 of epsilon. Each final point on each of the Fig. 5 curves corresponds to a single point in Fig. 3. Figure 5 shows that over the full time course, increasing epsilon leads to increased accuracy, at a higher cost.

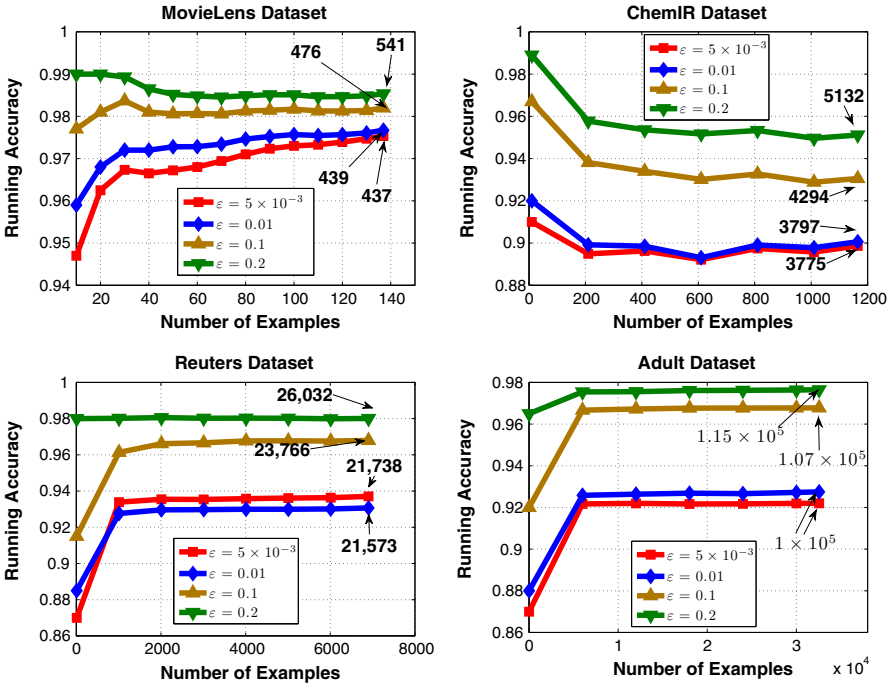


Fig. 5 Comparison of the running accuracy of CrowdSense at various ε values, averaged over 100 runs. In the plots, we show how many total votes were requested at each ε value

7.2 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 (5) 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 Fig. 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

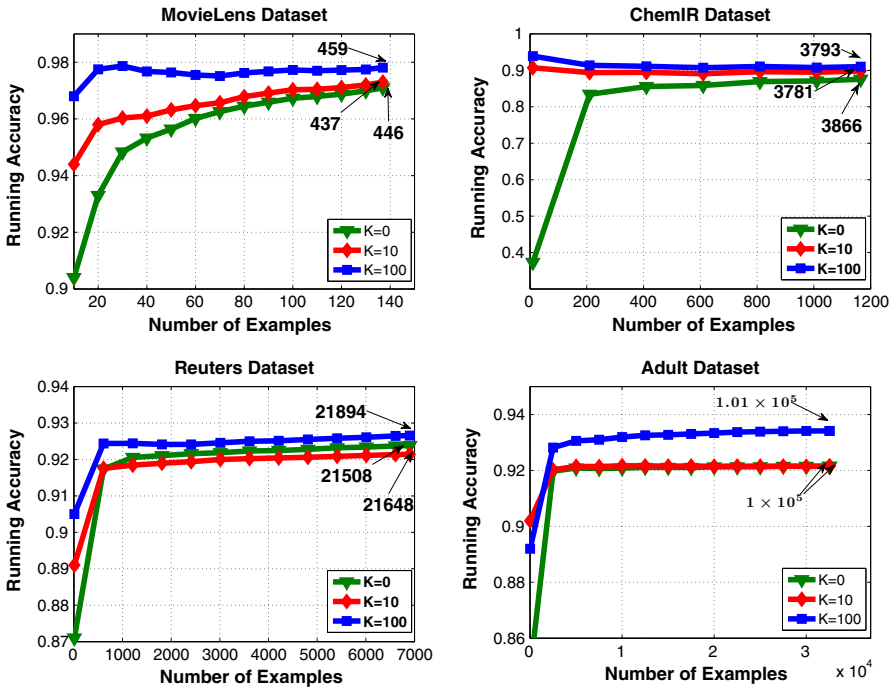


Fig. 6 Comparison of running accuracy with varying K. All curves use $\varepsilon = 0.005$, and are averaged over 100 runs

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.

8 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 smoothing done on the estimates of quality, discussed in Sect. 7.2,

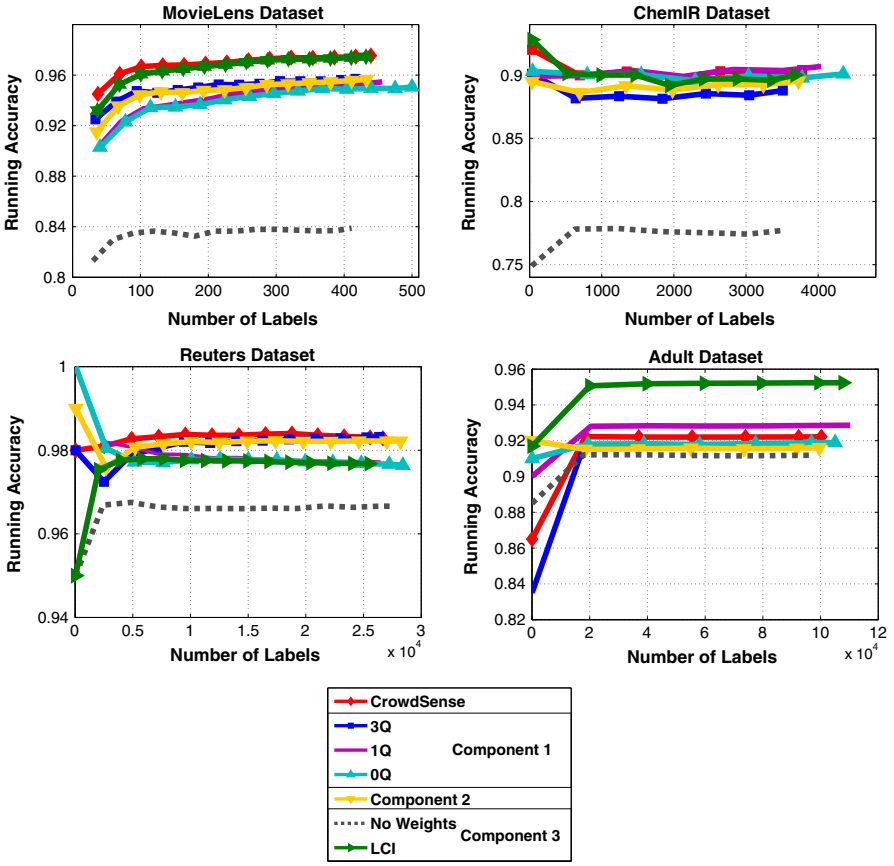


Fig. 7 Performance of CrowdSense and several of its variants, averaged over 100 runs. The x-axis is the total number of votes (the total cost), and the y-axis is the running accuracy on those examples. For the plots, we used $\varepsilon = 0.005$ for MovieLens, $\varepsilon = 0.01$ for ChemIR, $\varepsilon = 0.2$ for Reuters, and $\varepsilon = 0.1$ for Adult dataset

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 7 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 of CrowdSense was better. For the Adult dataset, 1Q achieves the highest final accuracy, but it also collects close to 10 K 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 Fig. 7 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 first 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.

We also experimented with a separate scoring scheme for estimating the labeler qualities. In particular, we estimated labeler i's quality at iteration t using a lower confidence interval. The lower confidence limit at level α, using the normal approximation to the binomial is:

$$Q_{it} = \hat{p}_{it} - z_{1-\frac{1}{2}\alpha}\sqrt{\frac{1}{c_i}\hat{p}_{it}(1-\hat{p}_{it})} \tag{8}$$

where $\hat{p}_{it} = \frac{a_{it}}{c_{it}}$ is the proportion of labeler i's labels that were consistent with the other labelers. The definition and computation of a and c were presented in Sect. 3. In Figure 7, the curve with the label LCI shows the running accuracy results for estimating the labelers' qualities using this scheme at $\alpha = 0.25$. The results are comparable to CrowdSense, performing similarly in some datasets, better in one, and worse than another, indicating that (8) is a reasonable alternative to the choice of labeler quality estimates for the weighting scheme.

9 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) smoothing 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 with respect to the majority vote. CrowdSense makes an implicit approximation on the joint probability distribution through its boosting-style update. We believe it is possible to further improve its accuracy by making an explicit approximation 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 approximations. These algorithms are designed to bring probabilistic insight into the workings of CrowdSense and other algorithms for approximating the crowd. The first variation (CrowdSense.Ind) makes a statistical independence approximation for the labelers. That is, we will assume that the majority vote is approximately independent of the votes that we have seen so far. This approximation is useful for large crowds, but may not be useful for smaller crowds. CrowdSense.Ind's probabilistic interpretation replaces 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 approximation, 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.

9.1 CrowdSense.Ind

CrowdSense.Ind assumes 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 can be 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 0.8, 0.5, and 0.5 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 approximation 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:

$$
\begin{aligned}
f(x_t|\text{votes}) = P(y=1|\underbrace{V_1,V_2,\ldots,V_i}_{\text{votes of labelers in } S}) &= \frac{\left(\prod_{l\in S}\psi_l\right)P_+}{\left(\prod_{l\in S}\psi_l\right)P_+ + \left(\prod_{l\in S}\theta_l\right)(1-P_+)} \\
&= \frac{1}{1+\dfrac{\left(\prod_{l\in S}\theta_l\right)(1-P_+)}{\left(\prod_{l\in S}\psi_l\right)P_+}} \qquad (9)
\end{aligned}
$$

where probabilities higher than 0.5 indicate a majority vote estimate of 1, and -1 otherwise. Further, the more the probability in (9) 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 B of the supplementary file, 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, 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

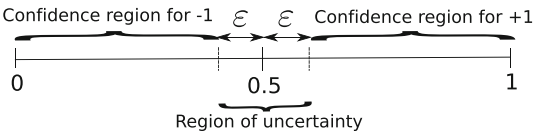


Fig. 8 The confidence intervals of the probabilistic approach with the independence approximation of the labelers

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

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

and expanding (9) to include the new hypothetical vote from the candidate labeler, we get

$$f(x_t|\text{votes}, V_{\text{candidate}}) = \frac{1}{1 + \frac{\left(\prod_{l \in S} \theta_l\right) \theta_{\text{candidate}} (1 - P_+)}{\left(\prod_{l \in S} \psi_l\right) \psi_{\text{candidate}} P_+}}. \tag{10}$$

The decision as to whether get a new vote depends on the values of Eqs. (9) and (10). 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 Fig. 8), then we simply make a decision and do not call any more votes. Pseudocode of CrowdSense.Ind is in Algorithm 2.

9.2 CrowdSense.Bin

The statistical independence approximation that we made in Sect. 9.1 is useful for sufficiently large crowds, but it is not a useful approximation 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 subcrowd 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 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:

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. **Initialize:** $a_{i1} \leftarrow 0$, $c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$ and $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$, $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$ random permutation of labeler id's
3. **Loop for** $t = 1, ..., N$
   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K}$, $i = 1, \ldots, M$. Update $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates. If quality estimates are identical, randomly permute labelers with identical qualities to attain an order.
   (b) 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 $\{3, \ldots, M\}$.
   (c) $V_{it}^{\text{bin}} = \frac{(V_{it}+1)}{2}, \quad \forall i \in S_t$
   (d) $\psi_{it} = Q_{it}^{V_{it}^{\text{bin}}} (1 - Q_{it})^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$
   (e) $\theta_{it} = (1 - Q_{it})^{V_{it}^{\text{bin}}} Q_{it}^{1-V_{it}^{\text{bin}}}, \quad \forall i \in S_t$
   (f) **Loop for** candidate $= 3 \ldots M$, candidate $\neq k$
      i. $f(x_t|\text{votes}) = \dfrac{1}{1 + \dfrac{(\prod_{i \in S_t} \theta_{it})(1 - P_+)}{(\prod_{i \in S_t} \psi_{it}) P_+}}$, where P_+ is the probability of a +1 majority vote.
      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.
   (g) $\hat{y}_t = 2 \times \mathbb{1}_{[f(x_t|\text{votes})>0.5]} - 1$
   (h) $a_{it} = a_{it} + 1, \quad \forall i \in S_t$ where $V_{it} = \hat{y}_t$
   (i) $c_{it} = c_{it} + 1, \quad \forall i \in S_t$
4. **End**

$$\begin{aligned} \text{Score} &= P_{X \sim \text{Bin}(\cdot, N_{\text{unvoted}}, 0.5)}(X \geq N_{\text{needed}}) - 0.5 \\ &= \left[\sum_{X=N_{\text{needed}}}^{N_{\text{unvoted}}} \text{Bin}(X, N_{\text{unvoted}}, 0.5) \right] - 0.5, \end{aligned} \tag{11}$$

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 (11) is always nonnegative; this is shown in Section A of the supplementary file. 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.

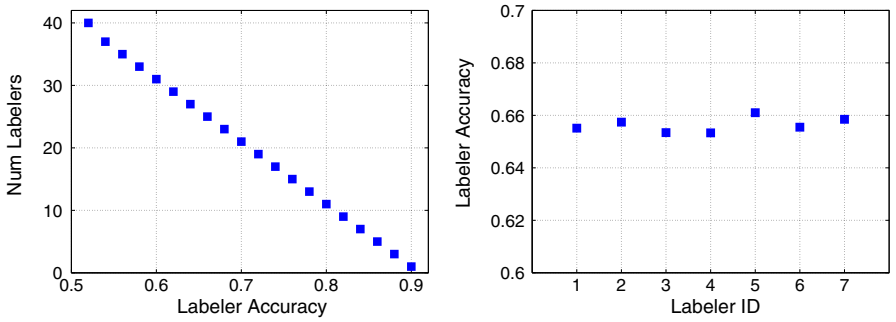
Algorithm 3 Pseudocode for CrowdSense.Bin

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. **Initialize:** $a_{i1} \leftarrow 0$, $c_{i1} \leftarrow 0$ for $i = 1, \ldots, M$ and $Q_{it} \leftarrow 0$ for $i = 1 \ldots M, t = 1 \ldots N$, $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$ random permutation of labeler id's
3. **For** $t = 1, ..., N$
   (a) Compute quality estimates $Q_{it} = \frac{a_{it}+K}{c_{it}+2K}$, $i = 1, \ldots, M$. Update $L_Q = \{l^{(1)}, \ldots, l^{(M)}\}$, labeler id's in descending order of their quality estimates. If quality estimates are identical, randomly permute labelers with identical qualities to attain an order.
   (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. $l_+ := \sum_{S_t} \mathbb{1}_{[V_{l_t}=1]}$ and $l_- := \sum_{S_t} \mathbb{1}_{[V_{l_t}=-1]}$
      ii. $l_{\text{current\_maj}} = \max(l_+, l_-)$
      iii. Majority_Is = $\left\lceil \frac{M}{2} \right\rceil$
      iv. Num_Needed_For_Majority = Majority_Is - $l_{\text{current\_maj}}$
      v. Score = $\left[\sum_{X=N_{\text{needed}}}^{N_{\text{unvoted}}} \text{Bin}(X, N_{\text{unvoted}}, 0.5)\right] - 0.5$
      vi. **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 $V_{it} = \hat{y}$, $a_{it} \leftarrow a_{it} + 1$
   (g) $\forall i \in S_t$, $c_{it} \leftarrow c_{it} + 1$
4. **End**

10 Comparative experiments with CrowdSense, CrowdSense.Ind and CrowdSense.Bin

We generated two datasets that simulate small and large crowds with synthetically generated labelers; a precedent for using synthetically generated "radiologist" labelers is provided by Raykar et al. (2010). For the large crowd dataset, we simulated labelers from the (50, 90 %] accuracy range. Starting from accuracy 90 % and one labeler, we reduce the required accuracy by 2 % while increasing the number of labelers to generate by 2. That is, we generate 3 labelers with accuracy 88 %, 5 labelers with accuracy 86 %, etc. for a total of 401 labelers. For the small crowd dataset, there are a total of 7 labelers with approximately equal qualities that are as low as possible. Since the labeler accuracies are determined by the majority vote of the crowd, it is not possible to have all labelers vote with accuracy very close to 50 %; the variance among the labeler accuracies reaches the minimum when the accuracies of the labelers are around 65 %. Both the small and large crowd datasets have 50,000 examples. The large crowd dataset enables us to assess the statistical independence approximation for the labelers in CrowdSense.Ind, whereas the small crowd dataset can help us validate setting a lower bound on how often the current subcrowd agrees with the majority vote in CrowdSense.Bin.

We show the breakdown of the labelers' accuracies in Fig. 9. We also present the tradeoff curves for CrowdSense, CrowdSense.Ind, CrowdSense.Bin and IE Thresh in Fig. 10. In most experiments, CrowdSense achieves better performance than CrowdSense.Ind and CrowdSense.Bin. On the other hand, the results on the simulated datasets show that CrowdSense.Ind and CrowdSense.Bin perform as well as, and



(a) N=50,000, M=401 **(b)** N=401, M=7

Fig. 9 Accuracy and labeler distribution of simulated datasets with large and small crowds. N and M denote the number of examples and the labelers in each dataset, respectively

sometimes better, than CrowdSense in simulated large and small crowds, respectively. In accordance with its derivation, the statistical independence approximation of CrowdSense.Ind can be valid for large crowds, but does not necessarily hold of small crowds. This behavior is observed in Fig. 10, where CrowdSense.Ind yields the highest accuracy for the large crowd, but performs poorly in the small crowd case. Conversely, CrowdSense.Bin mostly outperforms all other approaches in the small crowd case by favoring labelers with high accuracy, but lags the other CrowdSense variants on the large crowd dataset.

11 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. This problem appears frequently in *answer services* applications and polling. 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 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 approximation, and CrowdSense.Bin, where calculations are based on the binomial distribution. These two algorithms make probabilistic approximations or bounds on the joint distribution of the votes, and depending on the crowd they can potentially perform better than CrowdSense as shown in our experiments.

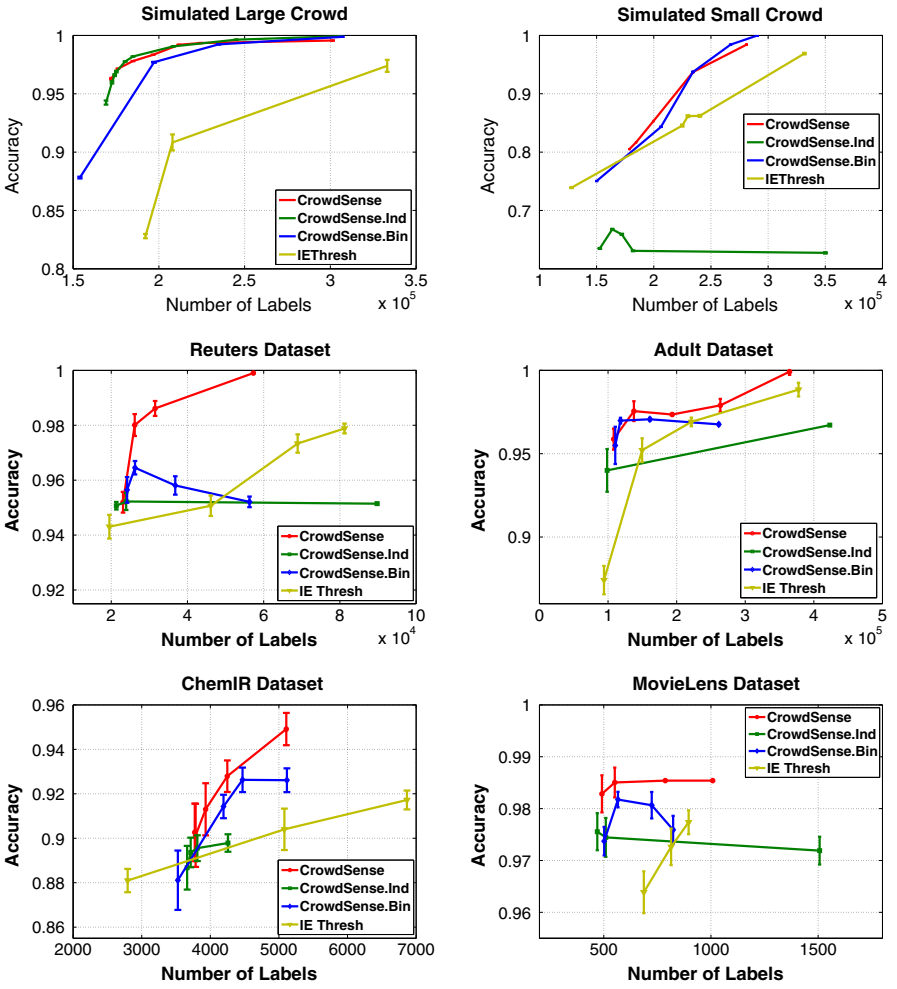


Fig. 10 Tradeoff curves for all datasets at comparable ε values. We used $K = 10$ for CrowdSense and variants, and initialized all algorithms with 4 examples as the gold standard data

References

Bernstein MS, Little G, Miller RC, Hartmann B, Ackerman MS, Karger DR, Crowell D, Panovich K (2010) Soylent: a word processor with a crowd inside. In: Proceedings of the 23rd annual ACM symposium on User interface software and technology (UIST), pp 313–322

Bernstein MS, Brandt J, Miller RC, Karger DR (2011) Crowds in two seconds: enabling realtime crowd-powered interfaces. In: Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST), pp 33–42

Bigham JP, Jayant C, Ji H, Little G, Miller A, Miller RC, Miller R, Tatarowicz A, White B, White S, Yeh T (2010) Vizwiz: Nearly real-time answers to visual questions. In: Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, ACM, New York, USA, UIST '10, pp 333–342

Callison-Burch C, Dredze M (2010) 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, pp 1–12

Dakka W, Ipeirotis PG (2008) Automatic extraction of useful facet hierarchies from text databases. In: Proceedings of the 24^{th} International Conference on Data Engineering (ICDE), pp 466–475

Dawid AP, Skene AM (1979) Maximum likelihood estimation of observer error-rates using the em algorithm. Appl. Stat. 28(1):20–28

Dekel O, Shamir O (2009a) Good learners for evil teachers. In: Proceedings of the 26^{th} Annual International Conference on Machine Learning (ICML)

Dekel O, Shamir O (2009b) Vox populi: Collecting high-quality labels from a crowd. In: Proceedings of the 22^{nd} Annual Conference on Learning Theory

Dekel O, Gentile C, Sridharan K (2010) Robust selective sampling from single and multiple teachers. In: The 23^{rd} Conference on Learning Theory (COLT), pp 346–358

Donmez P, Carbonell JG, Schneider J (2009) Efficiently learning the accuracy of labeling sources for selective sampling. In: Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining (KDD), pp 259–268

Downs JS, Holbrook MB, Sheng S, Cranor LF (2010) 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, pp 2399–2402

Ertekin S, Hirsh H, Rudin C (2012) Learning to predict the wisdom of crowds. In: Proceedings of Collective Intelligence, CI'12, Cambridge, Massachusetts

Gillick D, Liu Y (2010) 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, Association for Computational Linguistics, CSLDAMT '10, pp 148–151

Hsueh PY, Melville P, Sindhwani V (2009) Data quality from crowdsourcing: a study of annotation selection criteria. In: Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing, pp 27–35

Ipeirotis PG, Provost F, Wang J (2010) Quality management on amazon mechanical turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, ACM, New York, USA, HCOMP '10, pp 64–67

Ipeirotis PG, Provost F, Sheng VS, Wang J (2013) Repeated labeling using multiple noisy labelers. Data Min Knowl Discov 28(2):402–441

Kaisser M, Lowe J (2008) Creating a research collection of question answer sentence pairs with amazons mechanical turk. In: Proceedings of the 6^{th} International Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA)

Kasneci G, Gael JV, Stern D, Graepel T (2011) 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), pp 465–474

Law E, von Ahn L (2011) Human computation, synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, San Rafael

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

Mason W, Watts DJ (2009) Financial incentives and the "performance of crowds". In: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '09, pp 77–85

Nakov P (2008) Noun compound interpretation using paraphrasing verbs: Feasibility study. In: Proceedings of the 13^{th} international conference on Artificial Intelligence: Methodology, Systems, and Applications, Springer-Verlag, Berlin, Heidelberg, AIMSA '08, pp 103–117

Nowak S, Rüger S (2010) 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, pp 557–566

Ogawa S, Piller F (2006) Reducing the risks of new product development. MITSloan Manag Rev 47(2):65

Quinn AJ, Bederson BB (2011), Human computation: a survey and taxonomy of a growing field. In: Proceedings of the 2011 Conference on Human Factors in, Computing Systems, pp 1403–1412

Raykar VC, Yu S, Zhao LH, Valadez GH, Florin C, Bogoni L, Moy L (2010) Learning from crowds. J Mach Learn Res (JMLR) 11:1297–1322

Sheng VS, Provost F, Ipeirotis PG (2008) 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), pp 614–622

Smyth P, Burl MC, Fayyad UM, Perona P (1994a) Knowledge discovery in large image databases: Dealing with uncertainties in ground truth. In: KDD, Workshop, pp 109–120

Smyth P, Fayyad UM, Burl MC, Perona P, Baldi P (1994b) Inferring ground truth from subjective labelling of venus images. In: NIPS, pp 1085–1092

Snow R, O'Connor B, Jurafsky D, Ng AY (2008) 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), pp 254–263

Sorokin A, Forsyth D (2008) Utility data annotation with amazon mechanical turk. Computer Vision and Pattern Recognition Workshop 1–8

Sullivan EA (2010) A group effort: more companies are turning to the wisdom of the crowd to find ways to innovate. Mark News 44(2):22–28

Wallace BC, Small K, Brodley CE, Trikalinos TA (2011) Who should label what? instance allocation in multiple expert active learning. In: Proceedings of the SIAM International Conference on Data Mining (SDM)

Warfield SK, Zou KH, Wells WM (2004) Simultaneous truth and performance level estimation (staple): an algorithm for the validation of image segmentation. IEEE Transact Med Imaging (TMI) 23(7):21–903

Welinder P, Branson S, Belongie S, Perona P (2010) The multidimensional wisdom of crowds. In: Advances in Neural Information Processing Systems (NIPS) vol 10, pp 2424-2432

Whitehill J, Ruvolo P, fan Wu T, Bergsma J, Movellan J (2009) Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: Advances in Neural Information Processing Systems (NIPS), pp 2035–2043

Yan Y, Rosales R, Fung G, Dy J (2010b) Modeling multiple annotator expertise in the semi-supervised learning scenario. In: Proc. of the 26^{th} Conference on Uncertainty in Artificial Intelligence (UAI), AUAI Press, Corvallis, Oregon, pp 674–682

Yan Y, Rosales R, Fung G, Schmidt MW, Valadez GH, Bogoni L, Moy L, Dy JG (2010b) Modeling annotator expertise: Learning when everybody knows a bit of something. J Mac Learn Res-Proc Track 9:932–939

Zheng Y, Scott S, Deng K (2010) Active learning from multiple noisy labelers with varied costs. In: 10th IEEE International Conference on Data Mining (ICDM), pp 639–648