Efficient Crowdsourcing via Proxy Voting

Gal Cohensius, Omer Ben Porat, Reshef Meir, Ofra Amir

Technion—Israel Institute of Technology

Abstract

Crowdsourcing platforms offer a way to label data by aggregating answers of multiple unqualified workers. We introduce a *simple* and *budget efficient* crowdsourcing method named Proxy Crowdsourcing (PCS). PCS collects answers from two sets of workers: *leaders* (a.k.a proxies) and *followers*. Each leader completely answers the survey while each follower answers only a small subset of it. We then weigh every leader according to the number of followers to which his answer are closest, and aggregate the answers of the leaders using any standard aggregation method (e.g., Plurality for categorical labels or Mean for continuous labels). We compare empirically the performance of PCS to unweighted aggregation, keeping the total number of questions (the budget) fixed. We show that PCS improves the accuracy of aggregated answers across several datasets, both with categorical and continuous labels. Overall, our suggested method improves accuracy while being simple and easy to implement.

1 Introduction

Crowdsourcing is the process of completing tasks by soliciting contributions from a large group of people. In recent years, crowdsourcing has become increasingly prevalent, and is being used for a variety of tasks including data collection and labeling, prediction and open innovation. While crowdsourcing enables leveraging a large pool of workers, the quality of contributions strongly depends on workers' capabilities and motivation which varies widely across workers. In particular, since crowdsourcing workers are usually not experts and since some workers might not exert effort, answers obtained may be erroneous [13, 25, 26]. Thus, increasing the accuracy and efficiency of crowdsourced work is an important area of research [20, 1].

The overall quality of the crowdsourced task is determined by two factors: the quality of inputs (answers) provided by the individual crowd workers, and the method used to aggregate those answers. Prior work has examined ways to improve the quality of inputs (e.g., by providing performance-based monetary incentives [27]), as well as at methods for aggregating multi-source noisy estimations toward "truth discovery" [15, 10].

Aggregation of opinions is closely related to *social choice*, where the opinions or preferences of multiple voters are aggregated using some voting method. Mao et al. [18, 19] compared the performance of various voting rules for aggregating rankings, and showed that some of them yield an outcome that is significantly closer to the ground truth than others. More generally, Mao et al. argue that while sophisticated statistical methods have been developed for specific human computation tasks, these tools often require "*significant amount of data, work, knowledge, and computational resources,*" ([18], p. 141) which highlights the advantage of using simple voting rules.

In this paper we suggest to use a simple heuristic that is based on *proxy voting* to improve the accuracy of aggregated answers in crowdsourcing platforms. Proxy voting allows voters not to vote directly, and instead select a *proxy* (presumably another voter with similar opinions) to vote on their behalf. Several papers analyzed the benefits of proxy voting to society from theoretical and empirical aspects (see related work). In a recent theoretical paper, Cohensius et al. [3] showed that with proxy voting, the preferences of a large population from a given distribution can be represented well with only a small subset of "active voters". Therefore, we hypothesize that similar ideas can be applied to the problem of aggregating answers of crowd workers.

In this paper, we suggest a Proxy Crowdsourcing (PCS) policy, where answers are collected from two groups: *leaders* answer the complete set of questions, while *followers* are asked only a random

subset of the questions, which are only used to find the "closest" leader. Then, we aggregate the leaders' answers, where each leader is weighted by the number of followers to which she is closest.

The PCS method is not supposed to replace existing aggregation methods, but rather to boost their performance by properly weighing the workers or experts. For example, with PCS, instead of asking (say) 10 workers to answer 100 questions each and take the majority answers, we may ask only 5 workers (the leaders) to answer everything, and use the remaining budget to collect answers from 25 additional workers (the followers) who will answer 20 questions each. The aggregated answer is then calculated by taking the *Weighted Majority* of the 5 leaders.

As such, proxy crowdsourcing has potential to improve the accuracy of crowdsourcing while the total effort by workers remains fixed[1]. This paper examines the use of proxy crowdsourcing and identifies conditions under which PCS leads to higher quality outcomes (i.e., outcomes that are closer to the ground truth).

1.1 Related work

Improving the efficiency and quality of crowdsourcing The crowdsourcing literature has explored several avenues for improving the quality of workers' inputs. Some works focused on designing monetary incentives for workers to elicit higher quality inputs. For example, different schemes of providing workers bonus payments have been proposed and studied [29, 27, 28, 22], and peer prediction rules have been used in payment schemes to incentivize truthful reporting by workers [12, 7, 21]. Other works have suggested non-monetary incentives, such as engaging workers by increasing their curiosity [14] or eliciting commitments from workers to provide high quality inputs [6]. Our approach does not attempt to directly improve the quality of the input provided by workers, but rather aggregates inputs in a way that aims to assign higher weights to higher quality workers.

Another practice commonly used to increase the quality of crowdsourcing is to monitor the quality of workers' responses by implementing "gold questions" with known answers in order to filter workers that fail in those questions [30, 8]. To improve the effectiveness of this approach, Bragg et al. [2] proposed methods for determining the optimal amount of gold questions to include. Some works have further explored the use of planning approaches to dynamically determine when to collect additional inputs from workers based on confidence in the current solution [4, 11]. Proxy crowdsourcing aims to reduce the number of inputs required without requiring the use of sophisticated planning methods.

Research in crowdsourcing has also considered different methods of aggregating the responses obtained from individual workers to achieve higher quality results [10, 11, 23, 24, 17]. Most closely related to our approach, Venanzi et al. [24] identify "communities" of workers that are similar to each other when workers only provide answers to part of the question set, and use a Bayesian computation to aggregate the inputs from the different communities. In contrast to this approach, proxy crowdsourcing aims to determine a priori how to utilize a budget (i.e., how many answers to elicit from workers), and does not require complex estimation computations in the aggregation of workers' inputs.

Importantly, we note that the proxy crowdsourcing approach is complimentary to most of the other approaches for improving the quality and efficiency of crowdsourcing processes, and can be used in conjunction with these approaches.

For an overview of proxy voting in the context of *preference aggregation* (i.e., when there is no ground truth), see Cohensius et al. [3].

---

[1]Although we do not tackle this point explicitly, reducing the amount of feedback required from each voter has received some attention in the social choice literature (see, e.g. [16, 5]), as means to lower the cognitive burden on human participants.

1.2 Contribution

We evaluate PCS versus standard aggregation methods on a wide range of datasets in several domains: some synthetic, some collected for this paper, and some that were collected for different purposes by other researchers.

Our evaluation shows that for almost all datasets, it is beneficial to aggregate the crowdsourcing answers using PCS, in the sense that it decreases the expected distance between the aggregated answers and the ground truth. In some datasets the improvement reaches up to 32%.

We analyze the reasons for this improvement by looking at the initial distribution of workers' competence, and the distribution of leaders' weights after the aggregation. We establish a preliminary theoretical result that explains why more accurate answer vectors get higher weights.

2 Model

We denote by $[\![X]\!]$ the indicator variable of event X. $[k]$ is a shorthand for $\{1, \ldots, k\}$.

Let X^k be a k-dimensional Euclidean space, and d be a distance metric on X^k. Let $\mathcal{X}_\perp \stackrel{\text{def}}{=} \mathcal{X} \cup \perp$, where $\perp$ is a missing entry. Denote by $\mathcal{X}_\perp^k$ the space of *partial vectors* of $\mathcal{X}^k$. We shall make use of the tilde notation, i.e. $\tilde{x}$, to indicate that a vector is partial. Denote by $\perp(\tilde{x}), \top(\tilde{x}) \subseteq [k]$, respectively, the missing and valid entries of $\tilde{x} \in \mathcal{X}_\perp^k$.

We denote by $x^{(j)}$ the j-th entry of a vector x. Given a (partial) vector $\tilde{x}$ and subset of indices $A \subseteq \top(\tilde{x})$, we define $\tilde{x}(A) = (\tilde{x}^{(i)})_{i \in A} \in \mathcal{X}^{|A|}$. We extend the pseudo-distance function d to partial vectors by only considering joint entries:

$$d(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2) \stackrel{\text{def}}{=} d(\tilde{\boldsymbol{x}}_1(A), \tilde{\boldsymbol{x}}_2(A)),$$

where $A = \top(\tilde{\boldsymbol{x}}_1) \cap \top(\tilde{\boldsymbol{x}}_2)$.

Given a multi-set of (complete) vectors $S_M = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ and a partial vector $\tilde{x}$, we denote by $\text{NN}(S_M, \tilde{x})$ the multi-set of vectors in S_M that are the closest to $\tilde{x}$ according to d. Formally,

$$\text{NN}(S_M, \tilde{\boldsymbol{x}}) = \text{argmin}_{j \in M} \, d(\boldsymbol{x}_j, \tilde{\boldsymbol{x}}).$$

Aggregation with leaders An *aggregation rule* is a function $\boldsymbol{g} : \mathbb{R}^m \times \mathcal{X}^{k \cdot m} \to \mathcal{X}^k$. That is, a function that maps a set of m weighted complete vectors to a single vector $\hat{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{w}_M, S_M)$.[2]

An *instance* in the space $(\mathcal{X}^k, d)$ is a tuple $(S_M, \tilde{S}_N)$, where:

- $S_M = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ is a set of complete answers ("leaders");
- $\tilde{S}_N = \{\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n\}$ is a set of partial answers ("followers");

Given an instance and an aggregation rule $\boldsymbol{g}$, the aggregated answer vector is $\boldsymbol{g}(S_M, \tilde{S}_N, d) = \boldsymbol{g}(\boldsymbol{w}_M, S_M)$ where $w_j = 1 + \sum_{i \in N} \frac{[\![j \in \text{NN}(S_M, \tilde{\boldsymbol{x}}_i)]\!]}{|\text{NN}(S_M, \tilde{\boldsymbol{x}}_i)|}$. That is, the number of followers to which $\boldsymbol{x}_j$ is closest, where each follower divides her voting weight equally among all of her nearest neighbors.

Example 1. *Consider a space $(\mathcal{X}^k, d)$ where $\mathcal{X} = \{0, 1\}, k = 4$, and d is the Hamming distance. Let $\boldsymbol{g}$ be the Weighted Majority function. Consider*

$$S_M = \{(0,0,0,0), (0,1,0,1), (0,1,1,0), (1,1,1,1)\}, \tilde{S}_N = \{(\perp, 1, \perp, 1), (\perp, 0, 1, \perp)\}.$$

Then $NN(S_M, \tilde{\boldsymbol{x}}_1) = \{2, 4\}$ and $NN(S_M, \tilde{\boldsymbol{x}}_2) = \{1, 3, 4\}$. Thus, the weights are $w_1 = w_3 = 1\frac{1}{3}, w_2 = 1\frac{1}{2}, w_4 = 1\frac{5}{6}$. The aggregated answer is $\hat{\boldsymbol{x}} = (0, 1, 1, 1)$.

---

[2] More precisely, this is a class of functions, one for every m. We only use simple and neutral aggregation rules where the effect of weights is like duplicating voters. That is, $\boldsymbol{g}((w_1, \ldots, w_i + 1, \ldots, w_m), S_M) = \boldsymbol{g}((w_1, \ldots, w_i, 1, \ldots, w_m), (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_m))$.

Evaluation metrics Given a *ground truth* vector x^* and an aggregated answer vector $\hat{x}$, We call $d(\hat{x}, x^*)$ the *aggregated error*. For every individual worker j, we call $d(x_j, x^*)$ the *individual error* of worker j.

Crowdsourcing policies Consider a population $\mathcal{W}$ of workers, which is a distribution over complete answer vectors $\mathcal{X}^k$.

A *crowdsourcing policy with leaders* (or just *policy*) is a tuple (m, n, α, g, d), where m is the number of leaders; n is the number of followers; α is the fraction of answers that each follower should answer; g is an aggregation rule; and d is a distance metric.

A policy is used first to sample an instance from the population, and then to aggregate answers of this instance as follows. The policy samples m full vectors S_M from $\mathcal{W}$ and n partial vectors $\tilde{S}_N$ with $\lfloor \alpha k \rfloor$ valid entries (selected uniformly at random). Then, an aggregated answer $\hat{x} = g(S_M, \tilde{S}_N, d)$ is computed. Note that $\hat{x}$ is a random variable.

E.g., if we have unlimited budget, we can recruit and aggregate a large number of workers to reduce the error. However, we assume that the budget is limited, and that the required budget is linear in the total number of questions we ask.

To evaluate the benefit of using leaders, we define $PCS_{\alpha,\beta}(B)$ the policy that spends a total budget of B, where a fraction β from this budget is spent on followers who provide answers on αk of the questions.

Except when specified otherwise, the policies we will use to compare crowdsourcing with and without leaders are $PCS(B) = PCS_{0.2,0.33}(B)$ and $CS(B) = PCS_{0,0}(B)$, respectively.

For example, if $k = 20$ and B is sufficient for 1200 answers, then $CS(B)$ results in the policy $(m = 60, n = 0, \alpha = 0, g, d)$ (60 workers that answer all questions); whereas $PCS(B)$ results in the policy $(m = 40, n = 100, \alpha = 0.2, g, d)$. That is, 40 leaders will answer all 20 questions, and the remaining half of the budget will be spent on 100 followers who will answer $\lfloor \alpha k \rfloor = 4$ random questions each.

3 Empirical Methodology

We evaluated the use of proxy voting on datasets from three sources: 1. datasets we collected using Amazon Mechanical Turk; 2. existing datasets from [22]; 3. datasets we generated from simple distributions for specified parameters. Workers were randomly divided into leaders and followers.

These datasets included a variety of tasks differing in the types of questions asked (e.g., multiple choice questions, continuous assessments). We next describe these in more detail.

In the datasets we collected, participants were given short instructions,[3] then they had to answer $k = 25$ questions. We recruited participants through Amazon Mechanical Turk. We restricted participation to workers that had at least 50 assignments approved. We plant in each survey a simple question that can be easily answered by a anyone who understand the Instructions of the experiment (known as Gold Standards questions). Participants who answered correctly the gold standard question received a payment of $0.3. Participants did not receive bonuses for accuracy. The study protocol was approved by the Institutional Review Board at the Technion.

For each new or existing dataset, we mention below the total number of collected answer vectors. However we emphasize that unless specified otherwise, aggregation was always performed with the $CS(B)$ and $PCS(B)$ policies, where B is sufficient budget for exactly 12 complete answer vectors. For robustness, we tested each policy by sampling 5000 instances with replacements, using the entire dataset as our population $\mathcal{W}$. For each instance, we computed the distance from the aggregated result $\hat{x}$ to the ground truth x^* of the dataset, and averaged the aggregated error over all instances. We refer to the average expected error of a policy as the *loss*.
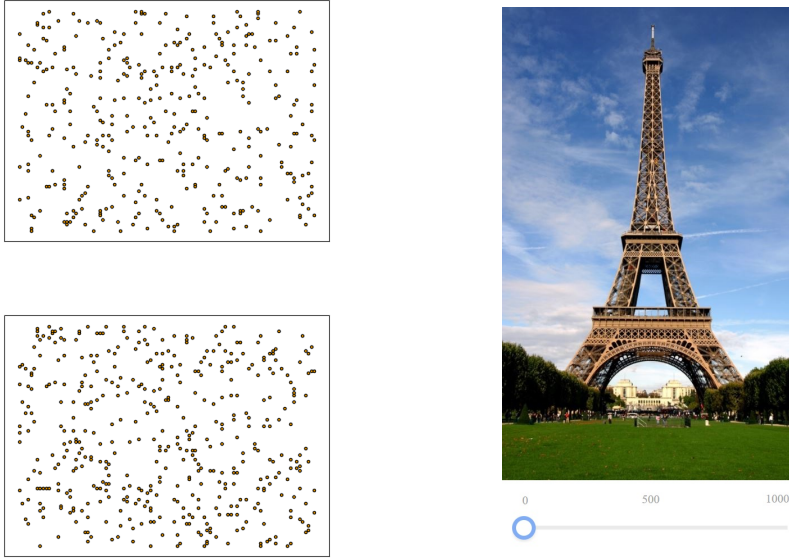
---

[3] see survey in https://goo.gl/W47X5R.

Figure 1: Representative questions of the Comparing dots (left) and Buildings' height (right) tasks. In the Comparing dots task, each worker was presented with a pair of dot figures, and had to determine which figure contained more dots. In the Buildings' height task, each worker was given a picture of a building, and had to estimate the building height in meters by selecting a number between 0 and 1000 along a slider.

3.1 Binary questions

Each question has two possible answers, $\mathcal{X} = \{a, b\}$. Hamming distance was used to calculate distances between vectors. Namely, the distance between two vectors is the number of questions on which they disagree. As an aggregation rule g we use Weighted Plurality, herein denoted by **wpl**. Formally,

$$\mathbf{wpl}(\boldsymbol{w}_M, S_m)^{(j)} \stackrel{\text{def}}{=} \begin{cases} a & \text{if } \sum_{i \in S_M : x_i^{(j)} = a} w_i > \frac{m}{2} \\ b & \text{if } \sum_{i \in S_M : x_i^{(j)} = b} w_i > \frac{m}{2} \end{cases},$$

and ties are broken uniformly at random. Note that in the binary case, Weighted Plurality and Weighted Majority coincide. The following datasets were examined:

Comparing Dots 41 workers were shown 25 pairs of pictures and had to estimate which picture out of the two has more dots[4]
Golden Gate 35 workers were asked 21 questions of identifying if the picture contains the Golden Gate Bridge. This dataset was collected by [22].

3.2 Categorical questions

Each question has a discrete set of answers, $\mathcal{X} = \{a, b, \dots, l\}$, and the distance function is the hamming distance. As in the Binary domain, we aggregate answers using **wpl** defined above when

---

[4]A similar task was performed in [19], where workers ordered a set of four pictures by increasing number of dots. While we used their dataset in a preliminary version of this paper, we found it unsuitable for the current paper since we wanted datasets where every participant makes more decisions.

domain $\mathcal{X}$	Task:	Workers	k (Questions)	Categories	$E[error]$	σ(error)
Binary	Comparing Dots	41	25	2	10.5	2.86
	Golden Gate [22]	35	21	2	4.7	3.4
Categorical	Heads of Countries[22]	32	20	4	1.8	2.6
	Flags[22]	35	126	4	47.8	29.6
	Dogs [22]	31	85	10	11.9	11.4
Continuous	Buildings' Height	208	25	1001	5271	3074
	Counting dots	201	25	1001	6159	1656

Table 1: Summary of datasets. For each dataset, we include the population size $|W|$, number of questions k, and the mean and variance of the individual error.

naturally extended to multiple possible answers. We used the following datasets, all collected by [22]:

Dogs 31 workers answered 85 questions, identifying the breed of the dog in the picture. We omitted 4 workers due to missing data in their report.
Heads of Countries 32 workers answered 20 questions, identifying heads of countries. We omitted 3 workers due to missing data in their report.
Flags 35 workers answered 126 questions, identifying countries to which flags belong.

3.3 Continuous questions

$\mathcal{X}$ is an interval $[0, H]$. The distance metric is the L^1-norm, thus $d(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j \leq k} |x^{(j)} - x'^{(j)}|$. We used the **Weighted Mean**, denoted by **wmn**, as an aggregation function[5]. Formally,

$$\mathbf{wmn}(\boldsymbol{w}_M, S_m)_j = \sum_{i \in M} w_i x_i^{(j)}.$$

We examined the following datasets:

Buildings' Height 208 workers were shown 25 buildings pictures, they were asked to estimate their height in meters. In Figure 1 we illustrate one such question.
Counting Dots 201 workers were shown 25 pictures of dots, they had to estimate the number of dots in each picture. Counting dots in images has been suggested as a benchmark task for human computation in [19, 9].

4 Analysis

Our main observation is that PCS improves accuracy of the aggregate answers by up to 30% as summarized in Table 2.

4.1 Why PCS improves accuracy

Possible (incorrect) explanation: sampling more workers reduces dependencies PCS sample more vectors, thus the dependency between answers decreases. We conjectured that this dependency

---

[5]We also considered the Weighted Median, which produced roughly the same results and hence omitted.

domain $\mathcal{X}$	Task:	$CS(12k)$	$PCS_{0.2,0.33}(12k)$	improvement [%]
Binary	Comparing dots	8.40	8.56	-1.90
	Golden gate [22]	2.16	1.88	12.96
Categorical	Heads of countries[22]	0.10	0.09	10.00
	Flags [22]	20.97	14.42	31.23
	Dogs [22]	2.98	2.69	9.73
Continuous	Buildings' height	3315	3075	7.24
	Counting Dots	5268.5	5188	1.53

Table 2: Results summary. Expected loss comparison across all our datasets between CS and PCS that spends $\beta = 1/3$ of the budget on followers and followers answers on $\alpha = 0.2$ of the questions in the survey. The budget B is sufficient for $12k$ answers in each domain. That is, 12 workers that answer on all the questions or 8 leaders and 40 followers.

decrease might reduce the loss. We compared CS to a policy were same budget is spent entirely on partial vectors. This method had the same loss as CS, therefore we rejected this explanation.

Better leaders get higher weight The improvement of PCS over CS is explained by the weights of leaders: we argue that better leaders receive higher expected weight. In Figure 2 (left), we used the policy $PCS_{0.2,0.375}(16k)$ which means we sample 10 leaders and 30 followers. We sampled workers from the Flags dataset. At each instance we sorted the leaders by their individual error and calculated their weights. The best leader got about twice the weight of worst leader. The observation that better leaders get higher weight is true across all of our datasets, but its magnitude differs. Another common observation is that the weakest leader gets higher weight than higher ranked leaders, the reason is that in some populations there is a group of weak workers (worse than random clickers), those weak workers delegate their voting weight with high probability to the weakest leader. Figure 2 (right) shows that weighting of the leaders improve the outcome substantially.
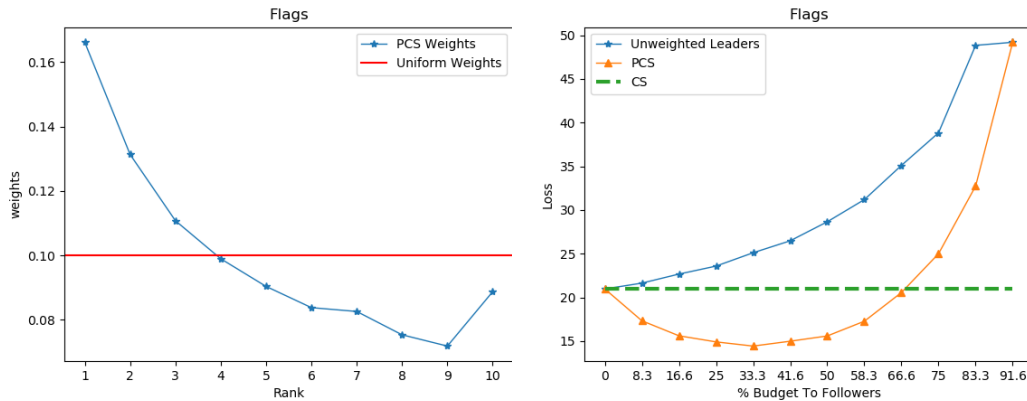

Figure 2: On the left, leaders voting weight sorted by their individual error. Averaged across 5000 instances, over the Flags dataset. Rank 1 means the leader with the lowest individual error in the sample, rank 10 means the leader with the highest individual error in the sample. On the right, the orange triangles shows the loss according to $PCS_{0.2,\beta}(12k)$, for reference, in blue dots, the loss for the same leaders when they are unweighted (only leaders, without followers).

4.2 Factors that effect PCS

Expected loss as a function of the policy   The expected loss depends on the policy used, that is, the way in which the budget is divided between obtaining leaders and followers. When increasing β (the fraction of the budget spent to on followers), the expected loss decreases until reaching an optimum. Further increasing β beyond this point results in higher loss compare to CS. The optimal β value depends on the dataset and ranges between $1/3$ and $1/2$. Higher β typically reduces accuracy because a sufficient number of leaders is needed for some high-quality leaders to be present, when some high-quality leader exists, adding followers is more cost effective.

To check this claim we evaluated the expected loss of $PCS_{0.2,\beta}(B)$ when using different β values and different budgets over all datasets. Figure 3 shows the expected loss generated by $CS(12k)$ and $PCS_{0.2,\beta}(12k)$ across different values of β on the Flags dataset. Results were averaged over 5000 instances (similarly to the results in the Table 2). The results are typical to most of the datasets and most budget values. Indeed, we see in Figure 3 that spending budget on followers is beneficial up to a certain amount (for this dataset and budget, the optimum is at $\beta = 1/3$).
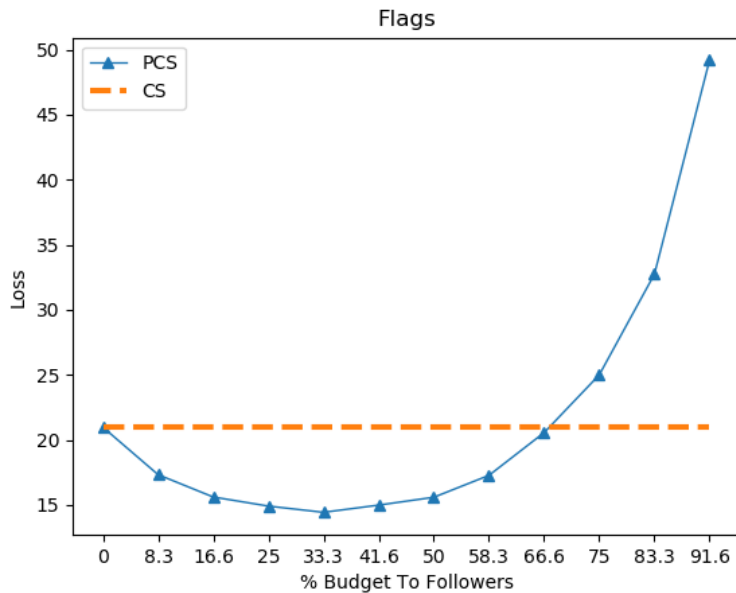

Figure 3: Expected loss as a function of β, the fraction of the budget spent to buy followers. The dashed line is the loss when using the CS policy (all the budget is used to pay for leaders). The continuous line uses $PCS_{0.2,\beta}(12k)$ (which means β of the budget is used to buy followers, and each follower is answering 0.2 of the questions.

Expected loss as a function of the datasets

1. **Domain**: **PCS is more effective in Categorical domain rather than Binary domain**. Table 2 provides anecdotal evidence for this claim. Intuitively, for PCS to work well, accurate workers must be close to other workers while inaccurate workers should be far from other workers. Consider two workers and one question and assume both workers provide a wrong answer for the question. In the binary domain their distance from each other is 0, while in the categorical domain they can be wrong in different ways so their distance can be 1.

2. **Number of questions: PCS is more effective in datasets with high number of questions**
   As the number of questions grows, the probability for incompetent workers to be wrong on
   the same question (with the same wrong answer) decreases, therefore the probability for a
   follower to follow incompetent leader decreases. See section 5 for a preliminary theorem. In
   Table 2 we see that the dataset with the highest number of question has the highest improve-
   ment rate of PCS.

3. **Individual error Variance: PCS is more effective when the variance of workers' Indi-
   vidual error is high.** When variance is high, some followers with high individual error can
   follow leaders with low individual error. When variance is low, followers will follow sim-
   ilar quality of leaders which will not improve the results. In order to verify this claim, we
   produced a synthetic population in the following manner: each worker has a competence pa-
   rameter $p_i \in (0, 1)$. A worker i with competence p_i answers each question correctly w.p. p_i.
   Errors are independent across questions and workers. The competence of each worker was
   sampled from a uniform distribution over an interval $[v_1, v_2]$, where v_1 and v_2 are parameters.
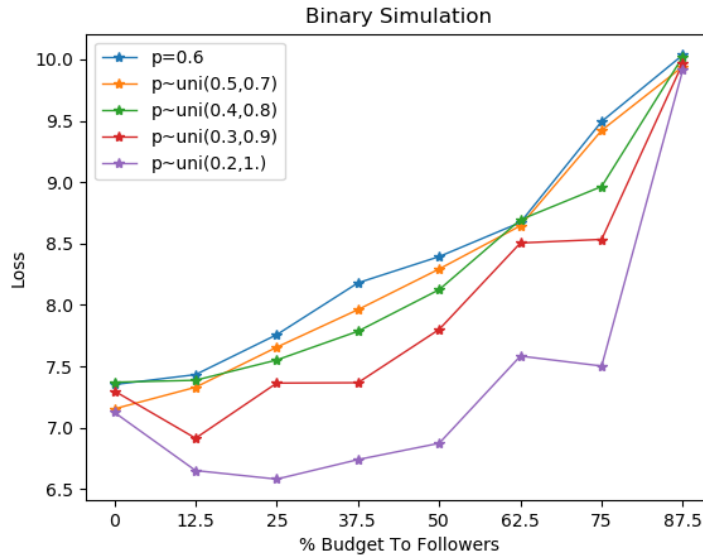

Figure 4: Loss as a function of β. Synthetic populations with similar mean but different variance of
their judgment competence p.

Figure 4 shows a comparison of populations that have similar competence means but differ
in their competence variance. PCS is only beneficial to populations with high variance (pur-
ple and red lines). Figure 5 present the individual errors distribution of the Flags and Dogs
datasets, the Flags dataset has a bimodal distribution, meaning that there are distinct groups of
"good" and "bad" workers. The Dogs dataset has a unimodal distribution. This might explain
why over the Flags dataset PCS reduces the loss better than over the Dogs dataset.

5 Explaining Leaders Weights

In the previous section, we observed empirically that competent leaders tend to get more followers
and thus higher weight. We are interested in a theoretical model that explains this phenomenon.
One such result was given by Cohensius et al. [3] for the limit case of infinite binary questions that
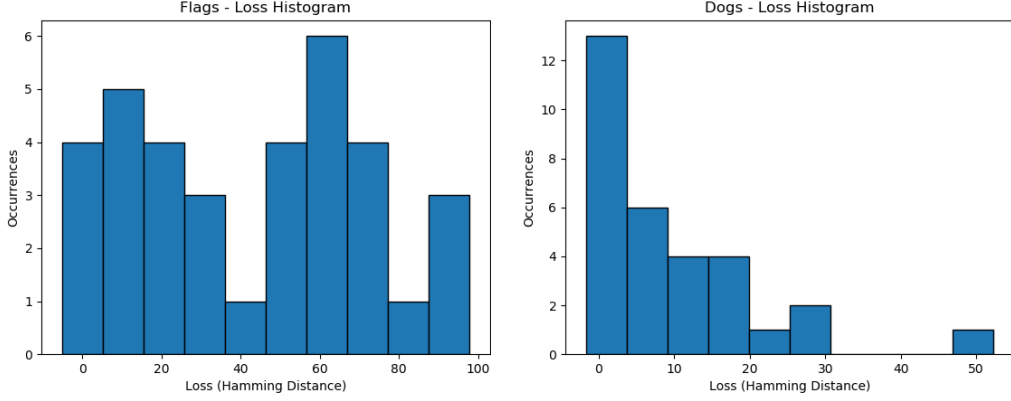
Figure 5: individuals errors histogram. On the left the Flags dataset shows bimodal distribution of individuals errors, Dogs on the right shows unimodal distribution of individuals errors.

are answered by followers $\alpha \cdot k \to \infty$. In this case, *all* followers select either the best or the worst leader, according to which one is closer. However, in realistic scenarios (including our datasets), the number of questions is typically dozens to hundreds.

We model a simplified version of the problem, where there is one follower which is requested to choose a leader amongst two possible leaders in a binary domain. Given a worker's competence, we only know the distribution of his answers, and we want to estimate the probability that the follower would choose the better leader.

Bounding the probability of a follower to choose the incompetent leader   Consider two leaders, High and Low, with judgment competence p_h, p_l respectively, such that $p_h > p_l > 0.5$. In addition, consider a follower Z with judgment competence $p_z > 0.5$. W.l.o.g. $\boldsymbol{x}^* = \mathbf{1}$. Thus $\boldsymbol{x}_h, \boldsymbol{x}_l$ and $\tilde{\boldsymbol{x}}_z$ are random binary vectors of length k. For simplicity, denote

$$\boldsymbol{s}_h = \boldsymbol{x}_h(\top(\tilde{\boldsymbol{x}}_z)), \boldsymbol{s}_l = \boldsymbol{x}_l(\top(\tilde{\boldsymbol{x}}_z)), \boldsymbol{s}_z = \tilde{\boldsymbol{x}}_z(\top(\tilde{\boldsymbol{x}}_z)).$$

That is, $\boldsymbol{s}_h, \boldsymbol{s}_l$ and $\boldsymbol{s}_z$ are the answers of High, Low and Z to the questions answered by Z. Hence $\boldsymbol{s}_h, \boldsymbol{s}_l$ and $\boldsymbol{s}_z$ are random binary vectors of length αk, whose entries are '1' with respective probabilities of p_h, p_l and p_z.

Lemma 2.
$$\Pr(d(\boldsymbol{s}_h, \boldsymbol{s}_z) \geq d(\boldsymbol{s}_l, \boldsymbol{s}_z)) \leq \exp\left(-\frac{\alpha k(p_h - p_l)^2(2p_z - 1)^2}{2}\right).$$

Proof. Define the random variable B_i such that

$$B_i = \begin{cases} 1 & \text{High and Z disagree, Low and Z agree} \\ -1 & \text{High and Z agree, Low and Z disagree} \\ 0 & \text{Otherwise} \end{cases}$$

where agreement/disagreement is with respect to the i-th question. Since the entries of $\boldsymbol{s}_h, \boldsymbol{s}_l$ and $\boldsymbol{s}_z$ are independent, so are B_i and $B_{i'}$ for $i \neq i'$. Notice that

$$\begin{aligned} \mathbb{E}(B_i) &= \mathbb{E}(B_i | s_z^{(i)} = 1) \Pr(s_z^{(i)} = 1) + \mathbb{E}(B_i | s_z^{(i)} = 0) \Pr(s_z^{(i)} = 0) \\ &= p_z \left((1 - p_h)p_l - p_h(1 - p_l)\right) + (1 - p_z)\left(p_h(1 - p_l) - (1 - p_h)p_l\right) \\ &= (2p_z - 1)(p_l - p_h). \end{aligned}$$

Notice that $d(\boldsymbol{s}_h, \boldsymbol{s}_z) - d(\boldsymbol{s}_l, \boldsymbol{s}_z) \sim \sum_{i=1}^{k} B_i$, therefore

$$\Pr(d(\boldsymbol{s}_h, \boldsymbol{s}_z) \geq d(\boldsymbol{s}_l, \boldsymbol{s}_z)) = \Pr(d(\boldsymbol{s}_h, \boldsymbol{s}_z) - d(\boldsymbol{s}_l, \boldsymbol{s}_z) \geq 0) = \Pr\left(\sum_{i=1}^{k} B_i \geq 0\right)$$

$$= \Pr\left(\frac{1}{k}\sum_{i=1}^{k} B_i - \mathbb{E}(B_i) \geq -\mathbb{E}(B_i)\right) \tag{1}$$

By invoking Lemma 3 on Equation (1), we obtain the desired result. $\qquad\square$

Lemma 3 (Hoeffding's Inequality). *Let $B_1 \ldots B_k$ be i.i.d. r.v. such that $B_i \in [-1, 1]$ and $\mathbb{E}(B_i) = \mu$. For every $t > 0$ it holds that*

$$\Pr\left(\frac{1}{k}\sum_{i=1}^{k} B_i - \mu \geq t\right) \leq \exp\left(-\frac{kt^2}{2}\right).$$

That is, the probability of Z selecting Low decreases exponentially in the distance between Low's and High's competence levels. Another observation is that if p_z approaches 0.5 (i.e. an incompetent follower), then the term in Lemma 2 approaches e^0, in other words, incompetent followers spread their weight roughly evenly over all leaders, whereas competent followers are substantially more likely to give their weight to a competent leader.

This supports the intuitive argument from [3] regarding the "Anna Karenina principle" (as good workers are indeed similar to one another), and thus at least partially explains the weight distribution of leaders.

6   Discussion

We introduced Proxy Crowdsourcing (PCS), an aggregation method that collects complete vectors and partial vectors of answers. The method then uses the partial vectors to weigh the complete vectors. We showed that this method can reduce the expected error in various crowdsourcing domains. PCS can be used in addition to other aggregation methods. Our conjecture is that for a wide range of aggregation methods, adding PCS to the aggregation method will improve the outcome. We observe that indeed this is the case for a wide variety of domains. Table 2 shows that PCS improves the outcome for $Plurality$ and $Mean$ aggregation rules. Similar results were obtained for the $Median$ rule.

Allowing Self Selection   In the experiment discussed in this paper we preallocated workers into leaders and followers. However preallocation is not mandatory, a requester can allow workers to choose after a sample of questions whether they wish to leave (followers) or to complete the survey (leaders). We hypothesize that allowing workers to *self select* their role can further improve the results. This can be done by using a bonus scheme that motivates the strong workers to become leaders and the weak workers to become followers. In preliminary experiments we saw that PCS with self selection produce more accurate outcome than self selection alone.

Allowing followers to choose their leader   Previous work [3] assumed that when given the option to choose their leader, followers follow the leader closest to them. In this work we wanted to test this assumption, by allowing followers to choose which leader they prefer out of a predefined set of leaders. We verified that indeed in crowdsourcing tasks, when workers are paid by the number of correct answers, followers tend to follow the leader closest to them. In an early setting of our experiment, followers were asked: "Why did you choose to follow this leader?" We observe two types of answers:

- The random follower: "I was unsure about my choices"

- The closest leader follower: "Choosing someone who guessed like I did would save time, and would likely give similar results."

This verifies our assumption that followers tend to follow the closest proxy.

References

[1] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.

[2] Jonathan Bragg, Daniel S Weld, et al. Optimal testing for crowd workers. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 966–974. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[3] Gal Cohensius, Shie Mannor, Reshef Meir, Eli Meirom, and Ariel Orda. Proxy voting for better outcomes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 858–866. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[4] Peng Dai, Christopher H Lin, Daniel S Weld, et al. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

[5] Joanna Drummond and Craig Boutilier. Elicitation and approximately stable matching with partial preferences. In *IJCAI*, pages 97–105, 2013.

[6] Avshalom Elmalech and Barbara J Grosz. " but you promised": Methods to improve crowd engagement in non-ground truth tasks. In *Fifth AAAI Conference on Human Computation and Crowdsourcing*, 2017.

[7] Xi Alice Gao, Andrew Mao, Yiling Chen, and Ryan Prescott Adams. Trick or treat: putting peer prediction to the test. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 507–524. ACM, 2014.

[8] Matthew R Gormley, Adam Gerber, Mary Harper, and Mark Dredze. Non-expert correction of automatically generated relation annotations. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 204–207. Association for Computational Linguistics, 2010.

[9] John J Horton. The dot-guessing game: A fruit flyfor human computation research. 2010.

[10] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 1–15. Springer, 2013.

[11] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[12] Ece Kamar and Eric Horvitz. Incentives for truthful reporting in crowdsourcing. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems-volume 3*, pages 1329–1330. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[13] Gabriella Kazai, Jaap Kamps, Marijn Koolen, and Natasa Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 205–214. ACM, 2011.

[14] Edith Law, Ming Yin, Joslin Goh, Kevin Chen, Michael A Terry, and Krzysztof Z Gajos. Curiosity killed the cat, but makes crowdwork better. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4098–4110. ACM, 2016.

[15] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. A survey on truth discovery. *ACM Sigkdd Explorations Newsletter*, 17(2):1–16, 2016.

[16] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *IJCAI*, volume 1, pages 287–293, 2011.

[17] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 745–754. ACM, 2015.

[18] Andrew Mao, Ariel D Procaccia, and Yiling Chen. Social choice for human computation. In *HCOMP-12: Proc. 4th Human Computation Workshop*, 2012.

[19] Andrew Mao, Ariel D Procaccia, and Yiling Chen. Better human computation through principled voting. In *AAAI*. Citeseer, 2013.

[20] Alexander J Quinn and Benjamin B Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1403–1412. ACM, 2011.

[21] Goran Radanovic, Boi Faltings, and Radu Jurca. Incentives for effort in crowdsourcing using the peer truth serum. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(4):48, 2016.

[22] Nihar Bhadresh Shah and Denny Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. In *Advances in neural information processing systems*, pages 1–9, 2015.

[23] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. In *Decision making and imperfection*, pages 1–35. Springer, 2013.

[24] Matteo Venanzi, John Guiver, Gabriella Kazai, Pushmeet Kohli, and Milad Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd international conference on World wide web*, pages 155–164. ACM, 2014.

[25] Jeroen Vuurens, Arjen P de Vries, and Carsten Eickhoff. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval (CIR11)*, pages 21–26, 2011.

[26] Paul Wais, Shivaram Lingamneni, Duncan Cook, Jason Fennell, Benjamin Goldenberg, Daniel Lubarov, David Marin, and Hari Simons. Towards building a high-quality workforce with mechanical turk. *Proceedings of computational social science and the wisdom of crowds (NIPS)*, pages 1–5, 2010.

[27] Ming Yin and Yiling Chen. Bonus or not? learn to reward in crowdsourcing. In *IJCAI*, pages 201–208, 2015.

[28] Ming Yin, Yiling Chen, and Yu-An Sun. The effects of performance-contingent financial incentives in online labor markets. In *AAAI*, 2013.

[29] Ming Yin, Yiling Chen, and Yu-An Sun. Monetary interventions in crowdsourcing task switching. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[30] Ce Zhang, Feng Niu, Christopher Ré, and Jude Shavlik. Big data versus the crowd: Looking for relationships in all the right places. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 825–834. Association for Computational Linguistics, 2012.

Gal Cohensius
Technion—Israel Institute of Technology
Technion, Israel
Email: `Gal.Cohensius@gmail.com`

Omer Ben - Porat
Technion—Israel Institute of Technology
Technion, Israel
Email: `omerbp@gmail.com`

Reshef Meir
Technion—Israel Institute of Technology
Technion, Israel
Email: `Reshef.Meir@ie.technion.ac.il`

Ofra Amir
Technion—Israel Institute of Technology
Technion, Israel
Email: `ofra.amir@gmail.com`