Crowd ideation of supervised learning problems

James P. Bagrow[1,2,*]

[1]Department of Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

February 14, 2018

Abstract Crowdsourcing is an important avenue for collecting machine learning data, but crowdsourcing can go beyond simple data collection by employing the creativity and wisdom of crowd workers. Yet crowd participants are unlikely to be experts in statistics or predictive modeling, and it is not clear how well non-experts can contribute creatively to the process of machine learning. Here we study an end-to-end crowdsourcing algorithm where groups of non-expert workers propose supervised learning problems, rank and categorize those problems, and then provide data to train predictive models on those problems. Problem proposal includes and extends feature engineering because workers propose the entire problem, not only the input features but also the target variable. We show that workers without machine learning experience can collectively construct useful datasets and that predictive models can be learned on these datasets. In our experiments, the problems proposed by workers covered a broad range of topics, from politics and current events to problems capturing health behavior, demographics, and more. Workers also favored questions showing positively correlated relationships, which has interesting implications given many supervised learning methods perform as well with strong negative correlations. Proper instructions are crucial for non-experts, so we also conducted a randomized trial to understand how different instructions may influence the types of problems proposed by workers. In general, shifting the focus of machine learning tasks from designing and training individual predictive models to problem proposal allows crowdsourcers to design requirements for problems of interest and then guide workers towards contributing to the most suitable problems.

Keywords— citizen science; novel data collection; Amazon Mechanical Turk; top-k ranking; randomized control trial

1 Introduction

We study how to combine crowd creativity with supervised learning. Crowdsourcers often use crowd workers for data collection and active learning, but it is less common to ask workers for creative input as part of their tasks. Indeed, crowdsourcing is typically not conducive to creative processes. Creativity is often enabled by autonomy and freedom [2], yet crowdsourcing, with its focus on the requirements of the crowdsourcer and the drive towards small microtasks, is generally not suitable for autonomy or freedom. This tension poses important crowdsourcing research challenges: what is the best and most efficient way to guide workers to creative ideation around a topic the crowdsourcer is interested in? How much guidance is possible without negatively impacting creativity? Can creative tasks be combined with traditional crowdsourcing tasks such as data collection?

In this paper, we investigate the following research questions:

1. Can crowd workers, who are typically not versed in the details of statistical or machine learning, propose meaningful supervised learning problems?

2. What is the best way to pose the task of proposing machine learning problems to workers, allowing them to understand the task of supervised learning while accounting for crowdsourcer requirements and minimizing potential bias? Does the design of the task influence what problems workers may propose?

3. Are workers able to compare and contrast previously proposed problems to help the crowdsourcer allocate workers towards problems deemed important or interesting?

4. What are the properties of problems proposed by workers? Do workers tend to ideate around certain topics or certain types of questions?

5. Can data collected for worker-proposed problems be used to build accurate predictive models without intervention from the crowdsourcer?

To study these questions, we implement and test a crowdsourcing algorithm where groups of workers ideate supervised learning problems, categorize and efficiently rank those problems according to criteria of interest, and collect training data for those problems. We study the topics and features of problems, show that performant predictive models can be trained on proposed problems, explore the design of the problem proposal task using a randomized trial, and discuss limitations and benefits when approaching learning from this perspective—how it changes the learning task from feature engineering and modeling to the specification of problem requirements.

The rest of this paper is organized as follows. In Sec. 2 we discuss crowdsourcing research on machine learning data collection and on creative ideation. In Sec. 3 we detail the end-to-end crowdsourcing algorithm we introduce for proposing supervised learning problems and generating novel datasets. Section 4 describes methods for how to rank proposed problems according to various criteria, saving crowdsourcer resources by eliminating poor problems and focusing data collection on problems of interest. In Sec. 5 we present our results applying our end-to-end algorithm using Amazon Mechanical Turk, studying features of problems proposed by workers and showing that supervised learning can be achieved on newly collected data. As crowdsourcing relies on worker's having a clear understanding of the task at hand, Section 6 describes a randomized trial we conducted to further understand how giving examples may help or hinder workers as they propose problems. Finally, we discuss our results and how our findings can inform future applications of crowdsourcing to machine learning problem ideation in Secs. 7 and 8.

2 Background

Crowdsourcing has long been used as an avenue to gather training data for machine learning methods [14]. In this setting, it is important to understand the quality of worker responses, to prevent gathering bad data and to maximize the wisdom-of-the-crowd effects without introducing bias [11]. Researchers have also studied active learning, where the supervised learning algorithm is coupled in a feedback loop with responses from the crowd. One example is the Galaxy Zoo project [12], where crowd workers are asked to classify photographs of galaxies while learning algorithms try to predict the galaxy classes and also manage quality by predicting the responses of individual workers.

While most crowdsourcing focuses on relatively rote tasks such as basic image classification [20], many researchers have studied how to incorporate crowdsourcing into creative tasks. Some examples include the work of Bernstein et al. [3] and Teevan et al. [24], both of which leverage crowd workers for prose writing; Kittur [13], where the crowd helps with translation of poetry; Chilton et al. [7], where the crowd develops taxonomic hierarchies; and Dontcheva et al. [8], where crowd workers were asked to ideate new and interesting applications or uses of common everyday objects, such as coins. In the context of machine learning, the website Kaggle provides a competition platform for *expert* crowd participants to create predictive models, allowing data holders to crowdsource modeling, but problems are posed by the data providers not the crowd.

Two recent studies take an approach to using crowdsourcing for creative ideation that share similarities with the algorithm we propose here. The work of Siangliulue et al. [22] studies crowdsourcing to write greeting cards, and implemented a proposal-and-ranking algorithm similar to our proposal-ranking-data-collection approach (Sec. 3). Similarly, the "Wiki Surveys" project [19] asks volunteers to contribute and vote on new ideas for improving quality-of-life in New York City. As with our algorithm, this project couples a proposal phase with a ranking and selection step, to create ideas and then filter and select the best ideas for the city government to consider. None of these studies applied crowdsourced creativity to problems of machine learning or data collection, however.

Perhaps the research most related to ours is the series of papers [4, 5, 23, 26]. That work studies the crowdsourcing of survey questions in multiple problem domains. Participants answered questions related to a quantity of interest to the crowdsourcer, such as how much they exercised vs. their obesity level or how much laundry they did at home compared with their home energy usage. Those participants were also offered the chance to propose new questions related to the quantity of interest (obesity level or home energy use). Supervised learning algorithms were deployed while crowdsourcing occurred to relate proposed questions (predictors) to the quantity of interest, and thus participants were performing crowdsourced feature engineering with the goal of discovering novel predictors of interest. Our work here generalizes this to crowdsourcing the entire supervised learning problem, not just the features, by allowing workers to propose not just questions related to a quantity of interest chosen by the research team, but also the quantity of interest itself.

3 Crowd ideation algorithm

Here we introduce a crowd ideation algorithm for the creation and data collection of supervised learning problems. The algorithm works in three phases: (i) problem proposal, (ii) problem selection by ranking, and (iii) data collection for selected problems. As part of the crowdsourcing, proposed problems may also be categorized or labeled by workers. This is an end-to-end algorithm in that crowd workers generate all problems and data without manual interventions from the crowdsourcer.

3.1 Problem proposal

In the first phase, a small number of workers are directed to propose sets of questions (see supplemental materials for the exact wording of these and all instructions we used in our experiments). Workers are instructed to provide a problem consisting of one *target question* and $p = 4$ *input questions*. We focused on four input questions here to keep the proposal task short; we discuss generalizing this in Sec. 7. Several examples of problems proposed by workers are shown in Table 1. Workers are told that our goal is to predict what a person's answer will be to the target question after only receiving answers to the input questions. Describing the problem in this manner allows workers to envision the underlying goal of the supervised learning problem without the need to discuss data matrices, response variables, predictors, or other field-specific vocabulary. Workers were also instructed to use their judgment and experience to determine "interesting and important" problems. Importantly, *no examples of questions were shown to workers*, to help ensure they were not biased in favor of the example (we investigate this bias with a randomized trial in Sec. 6). Workers were asked to write their questions into provided text fields, ending each with a question mark. They were also asked to categorize the type of answer expected for each question; for simplicity, we directed workers to provide questions whose answers were either numeric or true/false (Boolean), though this can be readily generalized. Lastly, workers in the first phase are also asked to provide answers to their own questions.

3.2 Problem ranking

In the second phase, new workers are shown previously proposed problems, along with instructions again describing the goal of predicting the target answer given the input answers, but these workers are asked to (i) rank the problems according to our criteria but using their own judgment, and (ii) answer survey questions describing the problems they were shown. It is useful to keep individual crowdsourcing tasks short, so it is generally too burdensome to ask each worker to rank all N problems. Instead, we suppose that workers will study either one problem or a pair of problems depending on the ranking procedure, complete the survey

Table 1: Examples of crowd-proposed supervised learning problems. Each problem is a set of questions, one target and p inputs, all generated by workers. Answers to input questions form the data matrix $\mathbf{X}$ and answers to the target question form the target vector y. Machine learning algorithms try to predict the value of the target given only responses to the inputs.

	Problem	Problem
Target:	**What is your annual income?**	**Do you have a good doctor?**
Input:	You have a job?	How many times have you had a physical in the last year?
Input:	How much do you make per hour?	How many times have you gone to the doctor in the past year?
Input:	How many hours do you work per week?	How much do you weigh?
Input:	How many weeks per year do you work?	Do you have high blood pressure?

	Problem
Target:	**Has racial profiling in America gone too far?**
Input:	Do you feel authorities should use race when determining who to give scrutiny to?
Input:	How many times have you been racially profiled?
Input:	Should laws be created to limit the use of racial profiling?
Input:	How many close friends of a race other than yourself do you have?

questions for the problem(s), and, if shown a pair of problems, to rate which of the two problems they believed was "better" according to the instructions given to them by the crowdsourcer. To use these ratings to develop a global ranking of problems from "best" to "worst", a crowdsourcer can apply top-K ranking algorithms such as those described below (Sec. 4). These algorithms select the K most suitable problems to pass along to phase three.

Problem categorization

As an optional part of phase two, data can be gathered to categorize what types of problems are being proposed, and what are the properties of those problems. To categorize problems a crowdsourcer can design any manner of survey questions depending on her interests, although it is helpful to keep the overall task short and as easy for workers to complete as possible. In our case, we asked workers what the topic of each problem is, whether questions in the problem were subjective or objective, how well the input questions would help to predict the answer to the target question, and what kind of responses other people would give to some questions. We describe the properties of proposed problems in Sec. 5.

3.3 Data collection and supervised learning

In phase three, workers were tasked with answering the input and target questions for the problems selected during the ranking phase. Workers could answer the questions in each selected problem only once but could work on multiple problems. In our case, we collected data from workers until each problem had responses from a fixed number of unique workers n, but a crowdsourcer could specify other criteria for collecting data. The answers workers give in this phase create the datasets to be used for learning. Specifically, the $n \times p$ matrix $\mathbf{X}$ consists of the n worker responses to the p input questions (we can also represent the answers to each input question i as a predictor vector x_i, with $\mathbf{X} = [x_1, \ldots, x_p]$). Likewise, the answers to the target question provide the supervising or target vector y.

After data collection, supervised learning methods can be applied to find the best predictive model $\hat{f}$ that relates y and $\mathbf{X}$, i.e., $y = \hat{f}(\mathbf{X})$. In our case, we focused on random forests [6], a powerful and general ensemble learning method. Random forests work well on both linear and nonlinear problems and can be used for both regression problems (where y is numeric) and classifications (where y is categorical). However, any supervised learning method can be applied in this context. For hyperparameters used to fit the forests, we chose 200 trees per forest, a split criterion of MSE for regression and Gini impurity for classification, and tree nodes are expanded until all leaves are pure or contain fewer than 2 samples.

4 Ranking proposed problems

Not all workers will propose meaningful problems, so it is important to include a ranking step (phase two) that filters out poor problems while promoting important problems. Furthermore, problems may not be meaningful for different reasons. Problems may lead to unimportant or unimpactful broader consequences, or problems may simply recapitulate known relationships (*"Do you think* $1 + 1 = 2$*?"*). Another reason is that they may lack *learnability*. For a binary classification task, learnability is reflected in the balance of class labels. For example, the target question "Do you think running and having a good diet are healthy?" is likely to lead to very many "true" responses and very few "false" responses. This makes learning difficult and not particularly meaningful (while a predictive model in such a scenario is not especially useful, the relationships and content of the target and input questions are likely to be meaningful, as we saw in some of our examples; see supplemental materials). Given these potential concerns, it is important to rank problems according to criteria of interest in order to focus the crowd on the most important and meaningful problems.

The choice of ranking criteria gives the crowdsourcer flexibility to guide workers in favor of, not necessarily specific types of problems, but problems that possess certain features or properties. This balances the needs of the crowdsourcer (and possible budget constraints) without restricting the free-form creative ideation of the crowd. Here we detail two methods to use crowd feedback to efficiently rank problems based on importance and learnability.

4.1 Importance ranking

We asked workers to use their judgment to estimate the "importance" of problems (see supplemental materials for the exact wording of instructions). To keep task size per work manageable, we assume workers only compare a pair of problems, asking them to compare two problems per task, with a simple "Which of these two problems is more important?"-style question. This reduces the worker's task to a pairwise comparison. Yet even reduced to pairwise comparisons, the global ranking problem is still challenging, as one needs $O(N^2)$ pairwise comparisons for N problems, comparing every problem to every other problem. Furthermore, importance is generally subjective, so we need the responses of many workers and cannot rely on a single response to a given pair of problems. Assuming we require L independent worker comparisons per pair, the number of worker responses required for problem ranking grows as $O(LN^2)$.

Thankfully, ranking algorithms can reduce this complexity. Instead of comparing all pairs of problems, these algorithms allow us to compare a subset of pairs to infer a latent score for each problem, then rank all problems according to these latent scores. For this work, we chose the following top-K spectral ranking algorithm, due to Negahban et al. [17], to rank crowd-proposed problems and extract the K best problems for subsequent crowdsourced data collection.

The algorithm begins with a *comparison graph* $G = (V, E)$, where the vertices $V = \{1, 2, \ldots, N\}$ denote the problems to be compared, and comparison between two problems i and j occurs only if $(i, j) \in E$. If any two problems are independently and equally likely to be chosen for comparison, then G is an Erdős-Rényi graph $G_{N,p}$ where p is the constant probability that an edge exists. In order to compute a global ranking from the pairwise comparisons, G must be connected. For Erdős-Rényi graphs, this occurs when $p \geq c \log(N)/N$

for some constant $c > 1$. The choice of an Erdős-Rényi comparison graph here is useful: when all possible edges are equally and independently probable, the number of samples needed to produce a consistent ranking is nearly optimal [17].

After generating a comparison graph, crowd workers are assigned ranking tasks for problem pairs $(i, j) \in E$. We seek L independent comparisons per edge, giving $L\,|E|$ total comparison tasks (less than the original $O(LN^2)$ when p is small). The outcome $t_{ij}^{(\ell)}$ of the ℓ-th comparison between problems i and j is:

$$t_{ij}^{(\ell)} = \begin{cases} 1, & \text{problem } j \text{ beats } i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

From Eq. (1), the aggregate comparison for pair (i, j) is $t_{ij} = \frac{1}{L}\sum_{\ell=1}^{L} t_{ij}^{(\ell)}$. Next, these t_{ij} are used to convert the comparison graph into a transition matrix $\mathbf{T}$ representing a first-order Markov chain. The elements of this transition matrix are:

$$T_{ij} = \begin{cases} \frac{1}{d} t_{ij}, & \text{if } (i, j) \in E, \\ 1 - \frac{1}{d}\sum_{k=1}^{N} t_{ik} A_{ik}, & \text{if } i = j, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where the constant $d \equiv \Delta(G)$ is the maximum vertex degree, $\mathbf{A} = [A_{ij}]$ is the adjacency matrix of G, and the diagonal terms T_{ii} ensure that $\mathbf{T}$ is a stochastic matrix. Lastly, the stationary distribution of this Markov chain, computed from the leading left eigenvector $\mathbf{u}$ of $\mathbf{T}$, provides the latent scores for ranking. The top-K problems then correspond with the K largest elements of $\mathbf{u}$.

For our specific crowdsourcing experiment, we generated $N = 50$ problems during the proposal phase, so here we generated a single Erdős-Rényi comparison graph of 50 nodes with $p = 1.5 \log(N)/N$, and opted for $L = 15$. Increasing L can improve ranking accuracy, but doing so comes at the cost of more worker time and crowdsourcer resources.

4.2 Learnability ranking

As discussed above, problems lack learnability when there is insufficient diversity in the dataset. If nearly every observation is identical, there is not enough "spread" of data for the supervised learning method to train upon. To avoid collecting data for such problems, we seek a means for workers to estimate for us the learnability of a proposed problem when shown the input and target questions. The challenge is providing workers with a task that is sufficiently simple for them to perform quickly yet the workers do not require training or background in how supervised learning works.

To address this challenge, we designed a task to ask workers about their opinions of the set of answers we would receive to a given question (a form of meta-knowledge). We limited ourselves to Boolean (true/false) target questions, although it is straightforward to generalize to regression problems (numeric target questions) by rephrasing the task slightly. Specifically, we asked workers what proportion of respondents would answer "true" to the given question. Workers gave a 1–5 Likert-scale response from (1) "No one will answer true" to (3) "About half will answer true" to (5) "Everyone will answer true". The idea is that, since a diversity of responses is generally necessary (but not sufficient) for (binary) learnability, classification problems that are balanced between two class labels are more likely to be learnable. To select problems, we use a simple ranking procedure to seeking questions with responses predominantly in the middle of the Likert scale. Specifically, if $t_{ij} \in \{1, \ldots, 5\}$ is the response of the i-th worker to problem j, we take the aggregate learnability ranking to be

$$t_j = \left| 3 - \frac{\sum_{i=1}^{W} t_{ij}\delta_{ij}}{\sum_{i=1}^{W} \delta_{ij}} \right|, \tag{3}$$

where W is the total number of workers participating in learnability ranking tasks, and $\delta_{ij} = 1$ if worker i ranked problem j, and zero otherwise. The closer a problem's score is to 3, the more the workers agree that target answers would be evenly split between true and false, and so we rank problems based on the absolute deviation from the middle score of 3. While Eq. (3) is specific to a 1–5 Likert scale variable, similar scores can be constructed for any ordinal variable.

This learnability ranking task can be combined with a pairwise comparison methodology like the one described for importance ranking. In our case, we elected to perform a simpler one-problem task because learnability ranking only requires examining the target question and because workers are less likely to need a relative baseline here as much as they may with importance ranking, where a contrast effect between two problems is useful for judging subjective values such as importance. Due to time and budget constraints we also took $K = 5$ for experiments using this ranking task.

Table 2: Summary of crowdsourcing tasks. Rewards are in USD.

Task	Reward	# responses	# workers
Problem proposal	\$3.00	50	50
Importance rating & problem categorization	\$0.25	2042	239
Learnability rating	\$0.05	835	83
Data collection for top importance problems	\$0.12	2004	495
Data collection for top learnability problems	\$0.12	990	281

5 Results

5.1 Crowdsourcing tasks

We performed crowdsourcing using Amazon Mechanical Turk during August 2017. Tasks were performed in sequence, first problem proposal (phase one), then ranking and categorization (phase two), then data collection (phase three). These tasks and the numbers of responses and numbers of workers involved in each task are detailed in Table 2, as are the rewards workers were given. Rewards were determined based on estimates of the difficulty or time spent on the problem, so proposing a problem had a much higher reward (\$3 USD) than providing data by answering the problem's questions (\$0.12 USD). No responses were filtered out at any point, although a small number of responses (less than 1%) were not successfully recorded.

We solicited $N = 50$ problems in phase one, compensating Mechanical Turk workers \$3 for their task. Workers could submit only one problem. A screenshot of the task interface for this phase (and all phases) is shown in the supplemental materials. Some example problems provided by crowd workers are shown in Table 1; all 50 problems are shown in the supplemental materials. After these problems were collected, phase two began where workers were asked to rate the problems by their importance and learnability and to categorize the features of the proposed problems. Workers were compensated \$0.25 per task in phase two and were limited to 25 tasks total. After the second phase completed, we chose the top-10 most important problems and the top-5 most learnable problems (Sec. 4) to pass on to data collection (phase three). We collected data for these problems until $n = 200$ responses were gathered for each problem (we have slightly less responses for some problems as a few responses were not recorded successfully; no worker responses were rejected). Workers in this phase could respond to more than one problem but only once to each problem.

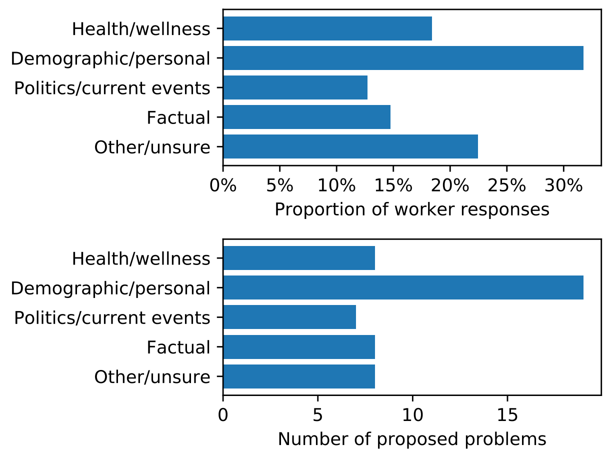


Figure 1: Categories of crowdsourced problems. The bottom plot counts the majority categorization of each problem.

5.2 Characteristics of proposed problems

We examined the properties of problems proposed by workers in phase one. We measured the prevalence of Boolean and numeric questions. In general, workers were significantly in favor of proposing Boolean questions over numeric questions. Of the $N = 50$ proposed problems, 34 were classifications (Boolean target question) and 16 were regressions (numeric target question). Further, of the 250 total questions provided across the $N = 50$ problems, 177 (70.8%) were Boolean and 73 were numeric (95% CI on the proportion of Boolean: 64.74% to 76.36%), indicating that workers were significantly in favor of Boolean questions over numeric. Likewise, we also found an association between whether the input questions were numeric or Boolean given the target question was numeric or Boolean. Specifically, we found that problems with a Boolean target question had on average 3.12 Boolean input questions out of 4 (median of 4 Boolean input questions), whereas problems with a numeric target question had 2.31 Boolean input questions on average (median of 2 Boolean input questions). The difference was significant (Mann-Whitney test: $U = 368.5, n_{\text{bool}} = 34, n_{\text{num}} = 16, p < 0.02$). Although it is difficult to draw a strong conclusion from this test given the amount of data we have (only $N = 50$ problems), the evidence we have indicates that workers tend to think of the same type of question for both the target and the inputs, despite the potential power of mixing the two types of questions.

To understand more properties of the questions workers proposed, we gave survey questions to workers as part of the importance rating task to categorize the problems shown to them. We used survey questions about

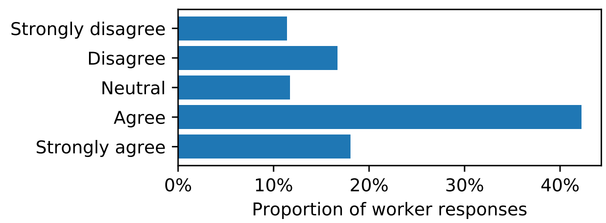


Figure 2: Worker responses to, “Are the input questions useful at predicting answers to the target question?”

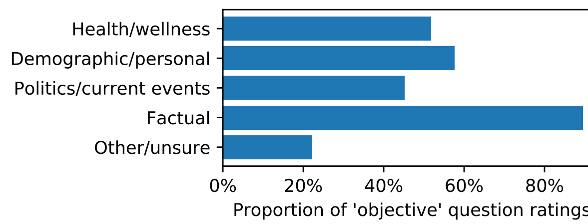


Figure 3: Proportion of question ratings of ‘objective’ instead of ‘subjective’ vs. the majority category of the problem.

the topical nature of the problem (Fig. 1), whether the inputs were useful at predicting the target (Fig. 2), and whether the questions were objective or subjective (Fig. 3). Problem categories (Fig. 1) were selected from a multiple choice categorization we determined manually. Problems about demographic or personal attributes were common, as were political and current events. Workers generally reported that the inputs were useful at predicting the target, either rating “agree” or “strongly agree” to that statement (Fig. 2). Many types of problems were mixes between objective and subjective questions, while problems categorized as “factual” tended to contain the most objective questions and problems categorized as “other/unsure” contained the most subjective questions, indicating a degree of meaningful consistency across the categorization survey questions.

To rank the learnability of classification problems, we asked workers about the diversity of responses they expected others to give to the Boolean target question, whether they believed most people would answer false to the target question, or answer true, or if people would be evenly split between true and false (Fig. 4). We found that generally there was a bias in favor of positive (true) responses to the target questions, but that workers felt that many questions would have responses to the target questions be split between true and false. This bias is potentially important for a crowdsourcer to consider when designing her own tasks, but seeing that most Boolean target questions are well split between true and false response also supports that workers are proposing useful problems; if the answer to the target question is always false, for example, then the input

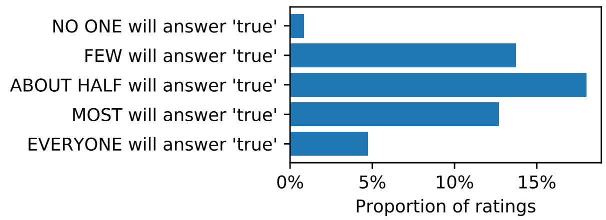


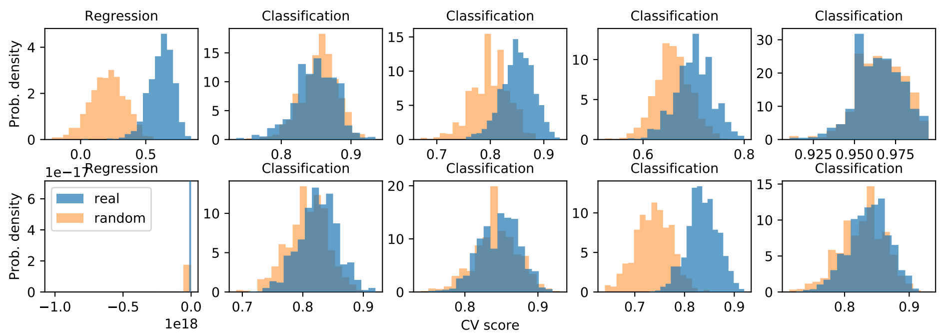
Figure 4: Crowd-categorized diversity of the (Boolean) target questions.

questions are likely not necessary, and the workers generally realize this when proposing their problems.

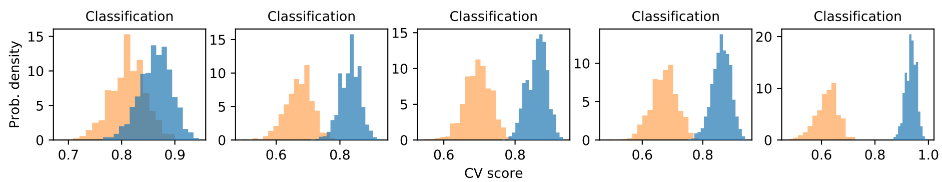
5.3 Performance of supervised learning on collected data

Given the proposed problems and the selection of problems for subsequent data collection, it is also important to quantify predictive model performance on these problems. Since workers are typically not familiar with supervised learning, there is a risk they may be unable to propose learnable problems. At the same time, however, workers may not be locked into traditional modes of thinking, such as assumptions that predictors are linearly related to the target, leading to problems with interesting and potentially unexpected combinations of predictor variables and the response variable.

Here we trained and measured the performance of random forest regressors and classifiers (Sec. 3.3), depending on whether the proposer flagged the target question as either numeric or Boolean, using the data collected for the 15 selected problems. Predictive performance was measured using the coefficient of determination for regressions and mean accuracy for classifications, as assessed with k-fold cross-validation (stratified k-fold cross-validation if the problem is a classification). To assess the variability of performance over different datasets, we used bootstrap replicates of the original crowd-collected data to estimate a distribution of cross-validation scores. There is also a risk of class imbalance, where nearly every target variable is equal and always guessing the majority class label can appear to perform well. To assess this, we also trained on a shuffled version of each problem's dataset, where we randomly permuted the rows of the data matrix $\mathbf{X}$, breaking the connection with the target variable y. If models trained on these data performed similarly to models trained on the real data, then it is difficult to conclude that learning has occurred, although this does not mean the questions are not meaningful, only that the data collected does not lead to useful predictive models.



(a) Top importance problems



(b) Top learnability (classification) problems

Figure 5: Cross-validation scores for (a) the top-10 importance ranked problems and (b) the top-5 learnable problems. At least two of the importance problems and four of the learnable problems demonstrate significant prediction performance. Performance variability was assessed with bootstrap replicates of the crowdsourced datasets and class imbalance was assessed by randomizing the target variable relative to the input data. Note that the second regression problem in panel a showed poor predictive performance for reasons we describe in Sec. 7.

The results of this model assessment procedure are shown in Fig. 5. Many of the 10 importance ranked problems in Fig. 5(a) demonstrate this class imbalance but at least two of the ten problems, one regression and one classification, show significant learning[1]. At the same time, four out of the five learnability-ranked problems (Fig. 5(b)) showed strong predictive performance.

These results show that, while many of the worker-proposed problems are difficult to learn on, it is possible to generate multiple problems where learning can be successful, and to assess this with an automatic procedure such as testing the differences of the distributions shown in Fig. 5.

5.3.1 Avoiding redundant questions

One concern when allowing non-experts to propose supervised learning problems is that they may introduce multiple questions that effectively ask the same thing. For example, it is redundant to ask both “Are you obese?” and “Is your BMI over 30?” within a single problem. This could lead to redundant predictors, which is inefficient, or redundant target variables, which is disastrous: if one can immediately guess the

[1] One regression problem showed terrible performance scores for reasons we detail in the discussion.

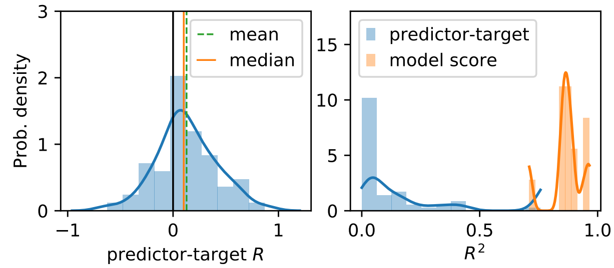


Figure 6: Distributions of correlations between individual predictors and the target variable, and the model score between the target variable and the predictions of the fitted model. Most problems do not feature a redundant predictor that can fully explain the target variable.

target answer given a predictor, then there is not much point in doing the supervised learning.

To determine if two natural language questions are equivalent short of rephrasing is a challenging computational linguistics task. However, with access to worker responses, we can infer redundancy based on correlations between the answers given during the data collection task. Therefore, to estimate redundancy, we measured the correlation between individual predictors x_i and the target variable y, $R(x_i, y)$, to determine any redundancy. We used the Pearson correlation coefficient which reduces to the phi coefficient if x_i and y are binary. The distribution of these predictor-target correlations for the 15 crowdsourced problems is shown in Fig. 6. Many predictors are only weakly correlated or anti-correlated with the target—on the right panel we see nearly all predictors have an $R^2 < 1/2$. In contrast, the model scores, the correlations between the observed y and the trained model $\hat{f}(X)$, also shown in the right panel, are all much higher. Together, these distributions imply that few if any target variables have redundant predictor questions, and thus workers generally avoided the concern of redundancy[2].

In Fig. 7 we compare the training and cross-validation scores for each problem as a function of the maximum predictor-target correlation for that problem ($\max_i R(x_i, y)$). If the learned model was only performing as well as the information available to it from the best single predictor, then the points would fall on the dashed line. However, we see that all problems[3] outperform that base single-predictor level, demonstrating that the problems workers proposed capture useful, non-redundant information.

[2]One can worry if this was due to workers simply answering the different questions randomly, but if that were the case it would be unlikely to have model scores as high as the ones we found in Fig 5.

[3]We excluded one regression problem from this set where learning was not possible. See the discussion for information on this.

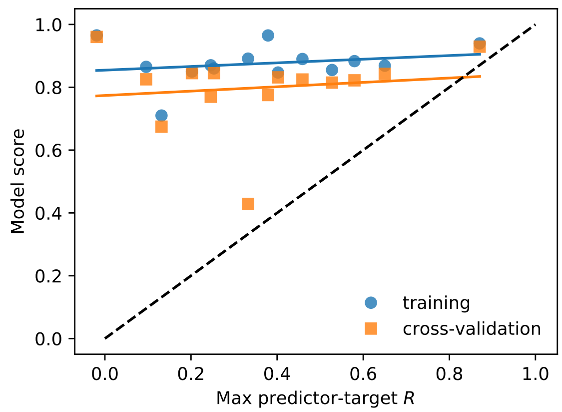


Figure 7: Predictive performance of learned models compared with the most correlated individual predictor for the problem. Under both training and cross-validation, learned models perform better than expected if there was a redundant predictor that perfectly or best explained the target (dashed line).

5.3.2 Positive correlation bias

While investigating the correlations between individual predictors and the target, we also found that positive correlations were more likely than negative correlations. The mean and median correlation over all predictor-target pairs, annotated in the left panel of Fig. 6, were significantly different from zero: mean correlation (95% CI) = 0.128 (0.0519, 0.204); median correlation (95% CI) = 0.101 (0.0420, 0.193). In other words, crowd workers were more likely to propose positive relationships that negative relationships, which has interesting implications given that in principle learning algorithms can perform just as well with both positive and negative relationships. This positivity bias was also observed by Wagy et al. [26], and leveraging it for improved response diversity, perhaps by focusing on workers who have a record of proposing anti-correlated relationships, is an interesting avenue for further research.

6 Task design: Do examples help explain the problem proposal task? Do examples introduce bias?

Care must be taken when instructing workers to perform the problem proposal task. Without experience in machine learning, they may be unable to follow instructions which are too vague or too reliant on machine learning terminology. Providing an example with the instructions is one way to make the task more clear while avoiding jargon. An example helps avoid the *ambiguity effect* [10], where workers are more likely to avoid the task because they do not understand it. However, there are potential downsides as well: introducing an example may lead to *anchoring* [25] where workers will be biased towards proposing problems related to the example and may not think of important, different problems.

To understand what role an example may play—positive or negative—in problem proposal, we conducted a randomized trial investigating the instructions given for the problem problem task. Workers who did not participate in previous tasks were randomly assigned to one of three arms when accepting the problem proposal task (simple random assignment). One arm had no example given with the instructions and was identical to the task studied previously (Sec. 5.2). The second arm included with the instructions an example related to obesity (*An example target question is: "Are you obese?"*), and the third arm presented an example related to personal finance (*An example target question is: "What is your current life savings?"*). The presence or absence of an example is the only difference across arms; all other instructions were identical and, crucially, workers were not instructed to propose problems related to any specific topic or area of interest.

After we collected new problems proposed by workers who participated in this randomized trial, we then initiated a followup problem categorization task (Sec. 3.2) identical to the categorization task discussed previously but with two exceptions: we asked workers to only look at one problem per task and we did not use a comparison graph as here we will not rank the problems for subsequent data collection. Since only one problem was categorized per task instead of two, workers were paid \$0.13 per task instead of the original \$0.25 per task. The results of this categorization task allow us to investigate the categories and features of the proposed problems and to see whether or not the problems differ across the three experimental arms.

6.1 Results

We collected $n = 90$ proposed problems across all three arms (27 in the no-example baseline arm, 33 in the obesity example arm, and 30 in the savings example arm), paying workers as before. We then collected 458 problem categorization ratings, gathering ratings from 5 or more distinct workers per proposed problem (no worker could rate more than 25 different problems). From these ratings we can study changes in problem category, usefulness of input questions at answering the target question, if the questions are answerable or unanswerable, and if the questions are objective or subjective, as judged by workers participating in the

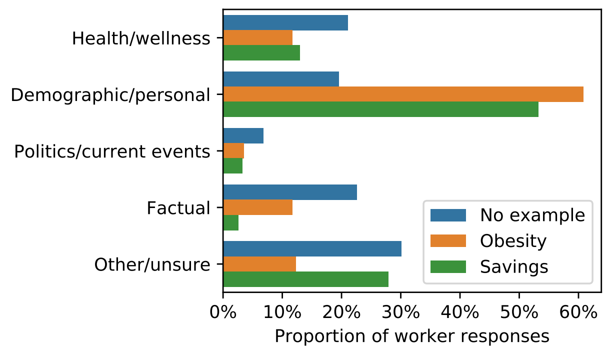


Figure 8: Categories of proposed problems under the different instructional treatments (the baseline instructions with no example, the instructions with the obesity example, and the instructions with the savings example). Problems proposed by workers who saw either example were more likely to be rated as demographic or personal and less likely to be considered factual. Interestingly, the obesity example led to fewer proposed problems related to health or wellness.

Table 3: Typical ratings and features of problems proposed under the three instruction types (the no-example baseline, the obesity example, and the savings example). Bold treatment quantities show a significant difference with the baseline (Table 4).

Rating or feature (variable type)	Mean x baseline	Mean x obesity	Mean x savings
Problem importance (x = 1–5 Likert; 5: Strongly agree)	3.09	3.16	**3.50**
Inputs are useful (x = 1–5 Likert; 5: Strongly agree)	3.36	**3.91**	3.67
Questions are answerable ($x = 1$) or unanswerable ($x = 0$)	0.82	**0.92**	**0.92**
Questions are objective ($x = 1$) or subjective ($x = 0$)	0.59	0.67	0.68
Questions are numeric ($x = 1$) or Boolean ($x = 0$)	0.25	**0.35**	**0.60**

followup rating tasks.

The results of this trial are summarized in Fig. 8 and Table 3, with statistical tests comparing the no-example baseline to the example treatments summarized in Table 4. In brief, we found that:

- Problem categories changed significantly across arms (Fig. 8 and Table 4), with more ‘demographic/personal’ problems, fewer ‘politics/current events’, and fewer ‘factual’ questions under the example treatments compared with the baseline.
- Workers shown the savings example were significantly more likely than workers in other arms to propose questions with numeric responses instead of Boolean responses: 60% of questions proposed in the savings arm were numeric compared with 25% in the no-example baseline ($p < 10^{-8}$; Table 4).

Table 4: Statistical tests comparing the categories and ratings given for problems generated under the no-example baseline with the categories and ratings of problems generated under the obesity example and savings example baselines. For categorical and Likert-scale ratings we used a Chi-squared test of independence while for binary ratings we used a Fisher exact test. Significant results ($p < 0.05$) are denoted with $*$.

		Baseline vs. obesity		Baseline vs. savings	
Difference in	Test	statistic	p-value	statistic	p-value
problem categories (cf. Fig. 8)	Chi-square	52.73$*$	$< 10^{-10}$	52.73$*$	$< 10^{-9}$
problem importance (cf. Table 3)	Chi-square	8.57	> 0.05	11.84$*$	< 0.02
inputs are useful (cf. Table 3)	Chi-square	16.35$*$	< 0.005	7.20	> 0.1
answerable/unanswerable (cf. Table 3)	Fisher exact	—$*$	0.0083	—$*$	0.012
objective/subjective (cf. Table 3)	Fisher exact	—	0.12	—	0.078
numeric/Boolean (cf. Table 3)	Fisher exact	—$*$	0.041	—$*$	$< 10^{-8}$

- All three arms were rated as having mostly answerable questions, with a higher proportion of answerable questions for both example treatments: 92% of ratings were 'answerable' for both example treatments compared with 82% for the baseline (Table 3). Proportions for both example treatments were significantly different from the baseline (Table 4).

- Workers more strongly agreed that the inputs were useful at predicting the target for problems proposed by workers under the example treatments than the no-example baseline problems. The overall increase was not dramatic however, and tested as significant ($p < 0.05$) only for the savings example vs. the baseline (Table 4).

- Questions proposed under the example treatments were more likely to be rated as objective than questions proposed under the no-example baseline: 67% and 68% of ratings were 'objective' for the obesity and savings examples, respectively, compared with 59% for the baseline (Table 3). However, this difference was not significant (Table 4).

Taken together, the results of this trial demonstrate that examples, while helping to explain the task to workers, will lead to significant changes in the features and content of problems the workers will propose. A crowdsourcer may be able to get better and somewhat more specific questions, depending on her requirements, but care should be taken when selecting which examples to use, as workers may anchor onto those examples in some ways when developing their own problem proposals.

7 Discussion

Here we studied a problem where crowd workers independently designed supervised learning problems and then collected data for selected problems. We determined that workers were able to propose learnable and important problems with minimal instruction, but that several challenges mean care should be taken when developing worker instructions as workers may propose trivial or “bad” problems. To avoid wasting resources on such bad problems, we introduced efficient problem selection and data collection algorithms to maximize the crowd’s ability to generate suitable problems while minimizing the resources required from the crowdsourcer.

Analyzing the problems proposed by workers, we found that workers tended to favor Boolean questions over numeric questions, that input questions tended to be positively correlated with target questions, and that many problems were related to demographic or personal attributes. To better understand how the design of the proposal task may affect the problems workers proposed, we also conducted a randomized trial comparing problems proposed by workers shown no example to those shown examples, and found that examples significantly altered the categories of proposed problems. These associations make it important to carefully consider the tasks assigned to workers, but they also provide opportunities to help the crowdsourcer. For example, it is less common for workers to mix Boolean and numeric questions, but workers that do propose such mixtures may be identified early on and then steered towards particular tasks, perhaps more difficult tasks. Likewise, given that examples have a powerful indirect effect on problem proposal, a crowdsourcer may be able to use examples to “nudge” workers in one direction while retaining more of their creativity than if they explicitly restricted workers to a particular type of problem. We saw an example of this in Sec. 6: workers shown the savings example were over 2.5 times more likely to propose numeric questions than workers shown no example.

When allowing creative contributions from the crowd, a challenge is that workers may propose trivial or uninteresting problems. This may happen intentionally, due to bad actors, or unintentionally, due to workers misunderstanding the goal of their task. Indeed, we encountered a number of such proposed problems, further underscoring the need for both careful instructions and the problem ranking phase. Yet, we found that the problem ranking phase did a reasonable job at detecting and down-ranking such problems, although there is room to improve on this further, for example by reputation modeling of workers or implementing other quality control measures [1, 14, 21]. More generally, it may be worth combining the ranking and data collection phases, collecting data immediately for all or most problems but simultaneously monitoring problems as data are collected for certain specifications and then dynamically allocating more incoming workers to the most suitable subset of problems [15, 16]. To monitor learnability, for example, a crowdsourcer can detect if most responses are similar or identical while data is collected, and deemphasize that particular problem accordingly.

The ranking procedures we used considered problem importance and problem learnability separately, but a single ranking may be more practical. One approach is to design the ranking task so that workers account for all the attributes the crowdsourcer wishes to select for at the same time. Another approach is to determine separate rankings for each desired attribute, then develop a joint ranking using a rank aggregation method such as Borda count [9]. Further work will be helpful in delineating which approaches work best for which problems.

We limited ourselves to numeric or Boolean questions, and a total of five questions per problem, but varying the numbers of questions and incorporating other answer types could be useful. For numeric questions, one important consideration is the choice of units. We encountered one regression problem (mentioned previously; see supplemental materials) where learning failed because the questions involved distances and volumes, but workers were not given information on units, leading to wildly varying answers. This teaches us that workers may need to be asked if units should be associated with the answers to numeric questions during problem ideation tasks.

One of the great potentials of crowdsourcing problem ideation is that it allows a diverse set of individuals to contribute their ideas and perspectives into the design of problems. Diversity can come in many forms, from gender and nationality to interdisciplinary training and education. Diversity is known to lead to a number of positive outcomes for groups and organizations [18] and *judicious* use of crowdsourced input may allow smaller teams of researchers to augment their own diversity with the crowd's diversity.

Allowing the crowd to propose problems changes the researcher's main workload from feature engineering and model fitting to problem specification. While here we allowed the crowd to ideate freely about problems, with the goal of determining what problems they were most likely to propose, in practice the crowdsourcer is likely to instead choose to focus on particular types of problems. As one example, a team of medical researchers or a team working at an insurance firm may request only problems focused on health care. Future work will investigating methods for steering the crowd towards topics of interest, in particular ways of focusing the crowd while biasing workers as little as possible.

8 Conclusion

In this work, we introduced an end-to-end algorithm allowing crowd workers to propose new supervised learning problems and to collect datasets on which to train predictive models. Ranking algorithms were used to eliminate poor problems and to select the problems most desirable to a crowdsourcer. We described the properties of the collected problems and validated the predictive performance of trained models. While not all proposed problems were useful or led to performant models, we demonstrated that the crowd can create multiple novel, learnable problems and then generate novel and useful datasets associated with those problems.

Acknowledgments This material is based upon work supported by the National Science Foundation under Grant No. IIS-1447634.

References

[1] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2): 76–81, 2013. 00152. 22

[2] T. M. Amabile, R. Conti, H. Coon, J. Lazenby, and M. Herron. Assessing the work environment for creativity. *Academy of management journal*, 39(5):1154–1184, 1996. 2

[3] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. *Communications of the ACM*, 58(8): 85–94, 2015. 00607. 3

[4] K. E. Bevelander, K. Kaipainen, R. Swain, S. Dohle, J. C. Bongard, P. D. H. Hines, and B. Wansink. Crowdsourcing Novel Childhood Predictors of Adult Obesity. *PLOS ONE*, 9(2):e87756, 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0087756. 00019. 4

[5] J. C. Bongard, P. D. Hines, D. Conger, P. Hurd, and Z. Lu. Crowdsourcing predictors of behavioral outcomes. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(1):176–185, 2013. 00024. 4

[6] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125, 1573-0565. doi: 10.1023/A:1010933404324. 7

[7] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1999–2008. ACM, 2013. 00117. 3

[8] M. Dontcheva, E. Gerber, and S. Lewis. Crowdsourcing and creativity. In *CHI 2011: Crowdsourcing Workshop*, 2011. 00010. 4

[9] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001. 22

[10] D. Ellsberg. Risk, ambiguity, and the Savage axioms. *The Quarterly Journal of Economics*, pages 643–669, 1961. 17

[11] P.-Y. Hsueh, P. Melville, and V. Sindhwani. Data quality from crowdsourcing: a study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 workshop on active learning for natural language processing*, pages 27–35. Association for Computational Linguistics, 2009. 3

[12] E. Kamar, S. Hacker, and E. 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, 2012. 3

[13] A. Kittur. Crowdsourcing, collaboration and creativity. *XRDS: crossroads, the ACM magazine for students*, 17(2):22–26, 2010. 3

[14] M. Lease. On quality control and machine learning in crowdsourcing. In *Proceedings of the 11th AAAI Conference on Human Computation*, AAAIWS'11-11, pages 97–102. AAAI Press, 2011. 3, 22

[15] Q. Li, F. Ma, J. Gao, L. Su, and C. J. Quinn. Crowdsourcing high quality labels with a tight budget. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 237–246. ACM, 2016. 00004. 22

[16] T. C. McAndrew, E. Guseva, and J. P. Bagrow. Reply & Supply: Efficient crowdsourcing when workers do more than answer questions. *PLOS ONE*, 12(8):e69829, 2017. doi: 10.1371/journal.pone.0182662. 22

[17] S. Negahban, S. Oh, and D. Shah. Rank centrality: Ranking from pairwise comparisons. *Operations Research*, 65(1):266–287, 2017. doi: 10.1287/opre.2016.1534. 8, 9

[18] S. E. Page. *The difference: How the power of diversity creates better groups, firms, schools, and societies*. Princeton University Press, 2008. 22

[19] M. J. Salganik and K. E. Levy. Wiki surveys: Open and quantifiable social data collection. *PLOS ONE*, 10(5):e0123483, 2015. 4

[20] E. Schenk and C. Guittard. Crowdsourcing: What can be outsourced to the crowd, and why. In *Workshop on Open Source Innovation, Strasbourg, France*, volume 72, 2009. 3

[21] F. Scholer, A. Turpin, and M. Sanderson. Quantifying test collection quality based on the consistency of relevance judgements. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1063–1072. ACM, 2011. 00053. 22

[22] P. Siangliulue, K. C. Arnold, K. Z. Gajos, and S. P. Dow. Toward collaborative ideation at scale: Leveraging ideas from others to generate more creative and diverse ideas. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 937–945. ACM, 2015. 4

[23] R. Swain, A. Berger, J. Bongard, and P. Hines. Participation and contribution in crowdsourced surveys. *PLOS ONE*, 10(4):e0120521, 2015. 00004. 4

[24] J. Teevan, S. T. Iqbal, and C. von Veh. Supporting Collaborative Writing with Microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2657–2668, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858108. 3

[25] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185(4157): 1124–1131, 1974. 18

[26] M. D. Wagy, J. C. Bongard, J. P. Bagrow, and P. D. H. Hines. Crowdsourcing predictors of residential electric energy usage. *IEEE Systems Journal*, PP(99):1–10, 2017. 4, 17