Human Computation (2024) 11:1:29-46

ISSN: 2330-8001, DOI: 10.15346/hc.v11i1

CrowdED: Statistical Guideline for Designing Optimal Crowdsourcing Experiments

AMRAPALI ZAVERI, Maastricht University

PEDRO V. HERNÁNDEZ SERRANO, Maastricht University

MICHEL DUMONTIER, Maastricht University

ABSTRACT

Crowdsourcing involves employing a large number of workers, creating HITs (Human Intelligent Tasks), submitting them to a crowdsourcing platform and providing a monetary reward for each HIT. One of the advantages of using crowdsourcing is that the tasks can be highly parallelized; that is, the work is performed by a high number of people in a decentralized setting. The design offers means to cross-check the accuracy of the answers by assigning each task to more than one person, thus relying on majority consensus and rewarding the workers according to their performance and productivity. However, since each worker is paid per task, the costs can significantly increase, irrespective of the overall accuracy of the results. Thus, one important question when designing such crowdsourcing tasks is whether we can estimate apriori - before launching the experiment - how many workers to employ and how many tasks to assign to each worker when dealing with large amounts of tasks. Therefore, we aim to answer the main research question: Can we a-priori estimate optimal workers and tasks' assignment to obtain maximum accuracy on all tasks?' We posit that realistic synthetic data generation can tackle these issues in practice. We introduce a two-staged statistical guideline, CrowdED, for optimal crowdsourcing experimental design to estimate optimal workers and task assignments through simulations to obtain maximum accuracy for crowdsourcing tasks. We describe the methodology, evaluate it considering real-world experiments, and show that the method performs better than a random selection of values.

1. CROWDSOURCING EXPERIMENTS

Crowdsourcing involves employing a large number of workers, creating HITs (Human Intelligent Tasks), submitting them to a crowdsourcing platform and providing a monetary reward for each HIT (Howe, 2006). The tasks primarily rely on basic human abilities and natural language

understanding but less on acquired skills such as domain knowledge. A significant share of the tasks addressed via microtask platforms like Amazon Mechanical Turk[1] (MTurk) could be called 'routine tasks' recognizing objects in images, transcribing audio and video material and text editing.

One of the advantages of using crowdsourcing is that the tasks can be highly parallelized; that is, the work is performed by a high number of workers in a decentralized setting. The design offers means to cross-check the accuracy of the answers by assigning each task to more than one person, thus relying on majority consensus and rewarding the workers according to their performance and productivity. However, since each worker is paid per task, the costs can significantly increase on a large scale, irrespective of the overall accuracy of the results. Thus, one important question when designing such crowdsourcing tasks is whether we can estimate, before launching an experiment, how many workers to employ and how many tasks to assign to each worker when dealing with large amounts of tasks. How do we optimally design the task so that the right combination of workers and tasks can produce the maximum accuracy, and can we determine this number apriori?

There have been studies that employ different methods such as active learning (Mozafari et al., 2014), test or gold standard questions to test worker aptitude (Hassan et al., 2016), self-reporting by workers of their knowledge or skills for the particular task (Ul Hassan et al., 2013) or on-the-fly optimization algorithms (Goel et al., 2017) for determining the optimal number of workers per task. However, reportedly, these are extremely expensive to adapt in a real-world experiment or only apply during or after executing a crowdsourcing experiment. This is where CrowdED contributes via apriori, through simulations, providing the optimal number of workers per task before launching the actual experiment, thus helping reduce costs. CrowdED, at the moment, only simulates Multiple Choice Questions (MCQs) type of tasks where the worker has to choose a single value from a given set of choices. In order to determine the number of workers and tasks that would be 'optimal' to solve the problem, we propose CrowdED, a two-staged Crowdsourcing Experimental Design. We aim to answer the main research questions: Can we a-priori estimate optimal workers and tasks' assignment to obtain maximum accuracy on all tasks?

The contributions of this paper are:
– 4 stand-alone modules that can be executed independently to simulate a crowdsourcing experiment
– CrowdED methodology and function to simulate the two-staged experiment
– evaluation of CrowdED via simulations and a real-world experiment
– open-source code, available as a Python library, executable in a Jupyter notebook[2]. The library builds on the *CrowdExperiments* code (Zaveri et al., 2018).

[1] https://www.mturk.com/
[2] https://www.github.com/MaastrichtU-IDS/crowdED

2. CROWDED METHODOLOGY

This section describes the rationale behind the simulation of a crowdsourcing experiment and details of the two-staged statistical design. Next, we describe the 5 stand-alone modules and the methodology in the following sections. In order to analyze the optimal parameters in a crowdsourcing experiment, a significant sample of tasks and many real-world experiments would be needed. However, this is expensive to perform on a large scale. Therefore, we propose generating a crowdsourcing experiment in a statistically reliable but synthetic way. In this way, the parameters can be analyzed via simulations, and ultimately, the proposed hypothesis can be tested. The algorithm is implemented in Python 3.7 and openly available 'crowdED'[3].

To ground our hypothesis on real-world experiments and determine default values in CrowdED, we chose parameters from 10 experiments available in online repositories[4] to perform simulations. These 10 experiments were selected because they represent multiple-choice questions (MCQ), the kind of experiments that CrowdED simulates. Secondly, the parameter (total no. of tasks, workers, no. of gold standard questions, workers per tasks, no. of categories) values also spanned a wide range of values (e.g., 200 tasks to 98000 tasks). Some experiments contained multiple microtasks. In those cases, we took only one. In some others, the overall accuracy was not reported. Thus, we calculated it using the gold standard tasks. We used these values to infer an informed accuracy value for an MCQ crowdsourcing experiment. This accuracy also calculates the alpha and beta parameters to feed into the CrowdED methodology as default values (as explained in the next section). Details of all the experiments are in Table 3.

2.1 Modules to Generate a Synthetic Crowdsourcing Experiment

The first problem to solve is to create an automated method to generate a synthetic experiment that reflects a real-world crowdsourcing experiment using the power of statistical modelling. To complete this task, we implemented 4 independent modules focusing on different functionalities and the CrowdED function that combines them, as depicted in Figure 1. The modules are:

– *Tasks Generation Module*: a module that generates a synthetic list of N desirable tasks
– *Workers Generation Module*: a module that generates a synthetic list of workers where each worker has a unique probability of carrying out a task; which is selecting one out of n elements in a multiple-choice question.
– *Task-Worker Assignment Module*: It assigns workers to tasks without tasks repetition
– Worker Inference Module: It infers the workers' answers and selects the best workers
– CrowdED function: a module that simulates the 2 stages experiment; takes in the requester's input values and generates the data frame with tasks, workers and worker answers along with each worker's performance and calculates the overall expected accuracy

[3] https://www.github.com/MaastrichtU-IDS/crowdsourcing-experiments
[4] https://www.data.world/crowdflower, https://www.figure-eight.com/data-for-everyone/, https://www.dbgroup.cs.tsinghua.edu.cn/ligl/crowddata/

We also explain the calculation of accuracy measures to assess the workers' performance and tasks. Note that the the crowdED[5] python library must be installed to run these modules.

To illustrate the modules, we use a running example from an existing crowdsourcing experiment of quality assessment of GEO metadata (Zaveri & Dumontier, 2017). The parameters for this experiment were: *total tasks = 1643, total workers = 145, workers per task = 3, number of categories = 9, and overall accuracy = 0.93*.

2.1.1 Tasks Generation Module

This module generates a synthetic list of tasks, where a task selects one out of *n* elements in a multiple-choice question. Given that there is no prior knowledge experiment's task nature, the probability that each task is answered correctly follows a *Uniform* distribution between *0* and *0.5* for hard tasks and *0.5* and *1* for easy tasks.

Hard Task ~ U (0, 0.5) *Easy Task ~ U (0.5, 1)*

Including the characterization of two types of difficulty (easy and hard) provides an additional parameter to the analysis that a requester can specify. However, this value is not mandatory; by default, the value is *0*, meaning that all tasks are easy. This module proposes that each simulated experiment be unique, for which the UUID library[6] is used to generate unique identifiers for each task randomly.

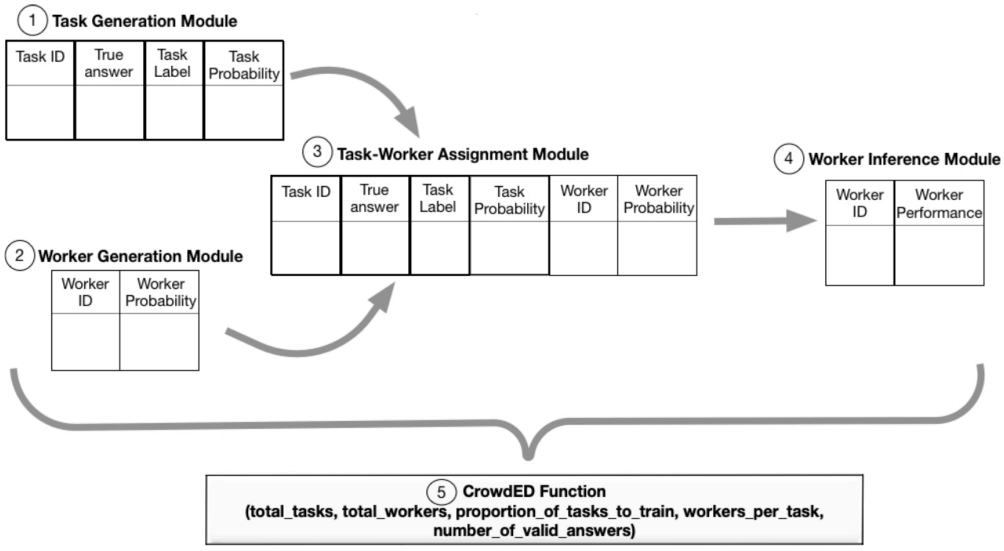

Figure 1. Overview of the four stand-alone modules and the CrowdED function.

[5] !pip install crowdED pycm shortuuid
[6] https://www.github.com/skorokithakis/shortuuid

The requester chooses the number of tasks desired corresponding to the experiment to simulate. The requester also has the option to select the number of valid answers for the experiment. However, this value must be odd. The module will choose terms from a random bag of words based on the number of correct answers. See, the categories generated at random are shown in Table 1.

bundles	data	generations
thirty	limitation	yarn
dentent	advertisement	materials

Table 1. ***Randomly generated nine valid answers.***

The module generates the desired number of tasks with unique identifiers, and in the same way generates the gold standard response for each task. This module can be executed by the following script. For example, Table 2 shows an output as a result of executing it.

```
import crowded.simulate as cs

#define your parameters
total_tasks = 415
p_hard_tasks = 0.4
number_of_valid_answers = 3

#create task dataset
df_tasks = cs.Tasks(number_of_valid_answers).create(total_tasks, p_hard_tasks)
```

task_id	true_answers	label_task	prob_task
task_E7xpdJTXNAQv	bundles	easy_task	0.89
task_L6qC6KcDnBca	data	easy_task	0.98
task_adu4Y8gyrqKP	bundles	easy_task	0.95
task_X6eJPEofGKf8	generations	easy_task	0.65
task_9eLzASnVhraq	generations	easy_task	0.59

Table 2. ***First 5 of 1643 rows of the Tasks Generation Module output with each task having a different probability score.***

2.1.2 Workers Generation Module

In this module, we calculate the probabilities of each worker getting an answer right. Since the workers' ability is unknown before the task is undertaken, we use a beta distribution to model the probability of the ability. It has been shown that the binomial beta model should be used for any experiment where humans have to answer an MCQ (SHAKIL, 2009). For example, this paper uses this binomial beta model specifically for crowdsourcing tasks to model the worker's ability (Fox, 2008). We reuse the probability density function (Goto et al., 2016). The probability density function $f(x|a,v)$ is given by:

$$f(x|a,v) = Beta\left(\frac{a}{\min(a,1-a)\,v}, \frac{(1-a)}{\min(a,1-a)\,v}\right)$$

Here $a \in (0,1)$ is the normalized value of the average ability of the workers in the crowdsourcing experiment; $v \in (0,1)$ is a parameter that determines the variance in worker ability. To determine the default values of alpha and beta, representing the worker distribution of an MCQ crowdsourcing experiment, we calculated them from the 10 real-world datasets listed in Table 3. Based on these values *(n=10)*, we calculated alpha = *accuracy * n = 0.76 * 10 = 7.6* and beta = *n*- alpha = *10 - 7.6 = 2.3*. Thus, the default values are alpha = *7.6* and beta = *2.3.*
The following script can execute this module. For example, Table 3 shows output as a result of running this module.

Dataset	Total no. of tasks	Total no. of workers	No. of gold standard questions	No. of workers per task	No. of categories	Overall accuracy
Fashion 10000 (Loni et al., 2014)	24457	1470	-	3	3	0.942
Sentiment Popularity (VENANZI et al., 2015a)	10000	5000	10000	20	2	0.893
Weather Sentiment (VENANZI et al., 2015b)	6000	300	6000	5	5	0.704
Query Document Relevance (Yilmaz et al., 2008)	98453	766	98453	5	2	0.408
Wikipedia image categorization (Wikipedia, 2019)	984	-	34	3	5	0.896
Company categorizations (crowdflower, 2019)	7335	-	183	3	6	0.818

Hate speech identification (Davidson et al., 2017)	24783	-	54	3	2	0.563
Dbpedia quality – datatypes (KIT, 2019a)	850	31	341	5	2	0.470
Dbpedia quality - object values (KIT, 2019c)	509	35	509	5	2	0.890
DBpedia quality – interlinks (KIT, 2019b)	223	31	223	5	2	0.940
					Mean	***0.760***

Table 3. ***Real-world crowdsourcing experimental values, which we used to determine the value of the beta parameters***

2.1.3 Task-Worker Assignment Module

This module ensures fair assignment of workers to the tasks listed, with the condition that no worker should repeat a task. To assign tasks, it starts by taking a task, and *k* workers are selected to answer that question, where k is the number of workers per task. Next, the workers are chosen randomly without replacement from the list of workers previously created, resulting in $\hat{k}$ different workers for each task. This process is repeated until the *N* number of tasks listed in the tasks table are assigned. Then, as a result of randomization, workers are given different tasks, just as in a real crowdsourcing experiment. These assignments will form a final table of size *kN*, i.e. number of workers per task *(k)* multiplied by the number of tasks *(N).* This module can be executed by the following script. For example, Table 4 shows an output as a result of executing this module.

```
import crowded.simulate as cs

#create task assignment
df_tw = cs.AssignTasks(df_tasks, df_workers, wpt).create()
```

task_id	worker_id	true_answers	label_task	prob_task	prob_worker
task_3GQs5ptNSUVU	7yLqPhe7TJty	yarn	easy_task	0.99	0.69135
task_3GQs5ptNSUVU	MnCuNgyMqzbT	yarn	easy_task	0.99	0.805523
task_3GQs5ptNSUVU	2bPFmZW8Pq6h	yarn	easy_task	0.99	0.697745

task_JDByzAtMpLhT	Ccv2aKc9r95p	thirty	easy_task	0.92	0.597253
task_JDByzAtMpLhT	RuiH3eBUzcHU	thirty	easy_task	0.92	0.676876

Table 4. ***First 5 rows of all combinations of the Task-Worker Assignment Module output, each worker having a different probability of getting the answer right, which is modeled using the binomial-beta distribution.***

2.1.4 Worker Inference Module

Inferring the workers' answer. In this module, the workers' answer is inferred using Bernoulli's probability density function using a prior conditional probability. Each combination in the resulting table from the worker generation module corresponds to a different probability for each event. Because these are related events, we proceed to use the conditional probability based on Bayes's theorem[7]. Let the random event *T* a worker to have the knowledge to answer a task, and the random event *w* a worker to answer correctly, then *P(w | T)*, is the conditional probability of a worker responding correctly given that she has the knowledge of answering the given task. The formula is based on the expression $P(B) = P(B|A)P(A) + P(B|A^c)P(A^c)$ which states that the probability of event *B* is the sum of the conditional probabilities of event *B* given that event *A* has or has not occurred. Bayes's formula for this case is defined as follows:

$$P(w|T) = \frac{P(T|w)P(w)}{P(T|w)P(w) + P(T|w^c)P(w^c)}$$

It is important to note that the probability $P(T|w^c)$ is the probability of the event of responding without knowing the answer. This probability corresponds to a random decision between the options. In a MCQ, this means that if a worker does not know the answer, the chances of answering a question correctly with *n* options is $P(T|w^c) = 1/n$.

The conditional probability is computed for the *kN* times. This probability of each combination serves as a prior probability to finding the answer that each worker will generate. Based on this logic, we define a random variable $X \sim Bernoulli(p)$ distributed, where p is the conditional probability explained above, given that each worker-task event has a different probability. Bernoulli's probability density function is calculated for all cases. As a result, each combination will be assigned a value of success or failure. Since we simulate MCQs for success, the value of the corresponding response is the same as that of the gold standard. In the case of failure, the

[7] https://plato.stanford.edu/entries/bayes-theorem/

corresponding value is a random selection of the $n - 1$ remaining valid answers. This module can be executed by the following script. For example, Table 5 shows an output as an execution result.

```
#compute probability
cp = cm.ComputeProbability(x, y, z)

#define the parameters
g = df_tw['true_answers'] #vector of gold standard answers
p = cp.predict() #binary vector of 0 and 1
z = df_tasks['true_answers'].unique()  #vector of valid answers in the experiment

#compute match
worker_answer = cm.WorkerAnswer(g, p, z)
#add the answers to the assignation dataset
df_tw['worker_answers'] = worker_answer.match()
```

task_id	worker_id	true answers	label task	prob task	prob worker	worker answers	inference
task_22By4uYxvA6u	hwY3hLriq7R4	materials	easy_task	0.95	0.996099	materials	1
task_22By4uYxvA6u	d36AYErUttLF	materials	easy_task	0.95	0.952213	materials	1
task_22By4uYxvA6u	HAiS5FFnBtQe	materials	easy_task	0.95	0.997918	materials	1
task_24aKhL7YT7hM	mYj9wgdDcPwC	yarn	easy_task	0.82	0.982841	yarn	1
task_24aKhL7YT7hM	6YYvoDyCwga6	yarn	easy_task	0.82	0.935455	generations	0

***Table 5.** First 5 row of the table assigns the best workers to the remaining tasks and computes the inference from the Worker Inference Module*

This module is essential for the quantitative analysis of the methodology since its objective is to calculate a performance measure for each worker and, in this way, select the best workers in a crowdsourcing experiment. It is essential to score the workers to demonstrate the hypothesis that trained workers on a portion of the tasks will obtain better accuracy. A workers' aggregation is carried out, summarizing the number of correct tasks and the number of tasks in total. The performance measure is the proportion of the correct answers over the total tasks. After this, a characterization is made where only those workers are selected with a measure of accuracy above average performance of all workers whose failures to respond correctly have been *0* or *1*. An example is shown in Table 6.

worker_id	prob_worker
2XYfi2sDkupf	0.965502
2bPFmZW8Pq6h	0.984682
YXBKkNZaLPP8	0.995149
WhEZYPp57f7m	0.990698
UwyUN7ufTkfh	0.997898

Table 6. ***First 5 rows of the table with the best workers after assessing the performance***

2.1.5 Accuracy Calculation

After the entire experiment is simulated along with the worker's answers, the next step is calculating the accuracy of the tasks. To do this calculation, we use the PyCM library (Haghighi et al., 2018) . PyCM[8] is a multi-class confusion matrix library written in Python that supports input data vectors and direct matrix and a tool for post-classification model evaluation that supports most classes and overall statistics parameters. PyCM is the swiss-army knife of confusion matrices with a broad array of metrics for predictive models and an accurate evaluation of a large variety of classifiers. As a result of this module, the overall accuracy of the tasks is obtained. The following script can execute the evaluation

```
from pycm import *
g = df_tw['true_answers'] #vector of gold answers
a = df_tw['worker_answers'] #vector of inferred answers

#compute confusion matrix
cm = ConfusionMatrix(list(g), list(a))
print(cm.Overall_ACC)
    > 0.9395
```

The accuracy computed with the CrowdED module is close to the actual accuracy of the real-world experiment 0.93.

2.2 **The CrowdED function**

The previously described modules are combined in the CrowdED methodology, which consists of 2 Stages. In Stage 1, the requester has the option to configure the following variables that represent her assumptions: No. of tasks, No. of workers, Proportion of hard tasks (optional, default 0, meaning all are easy tasks), Proportion of tasks to train, No. Of workers per task, the number of valid answers and alpha (optional) and beta (optional).

[8] https://www.github.com/sepandhaghighi/pycm

Then, based on the requester's input values, the data frame with tasks, workers and worker answers are simulated, and each worker's performance and overall accuracy are calculated.
This method aims to produce an entire experiment combining the 5 modules discussed in one function. The experiment's output is a table with the results, similar to a results table in any crowdsourcing platform.

At the end of Stage 1, we get:
– Poor workers, those that did not achieve high consensus amongst other workers performing the same task
– These workers are flagged and not chosen for Stage 2
– Best workers are those with a good performance and are assigned the best worker status
– These workers are selected for Stage 2
– The total number of tasks minus the Proportion of tasks to train
– These unassigned tasks are assigned to the best workers

In Stage 2, the best workers get assigned the unassigned tasks, the workers' answers are simulated, and the task accuracy is calculated. Finally, the tasks from Stages 1 and 2 are merged, and the accuracy of all the tasks is calculated.

For example, considering the sample dataset generated earlier, we assess the performance and get the 'best workers'. Next, we assign the 'best' workers to the remaining tasks. Then, we compute the workers' performance on the rest of the tasks. Finally, we calculate the accuracy of the simulated experiment at the end of the 2 stages, which is *98.01%* (higher than the accuracy obtained in Stage 1 and the real-world experiment of *0.83*).

3. EXPERIMENTS AND RESULTS

We evaluated our methodology by re-executing two existing real-world experiments in the Amazon Mechanical Turk (MTurk) crowdsourcing platform using CrowdED's two-stage approach. These experiments were chosen since they differ in task nature, the total number of tasks, and the overall accuracy obtained. The values for each of the variables for the two experiments are reported in Table 7. The two experiments are:

– *MetaCrowd:* This crowdsourcing experiment assesses the biomedical metadata quality of the Gene Expression Omnibus (GEO) dataset (Zaveri and Dumontier, 2017). The *1643* tasks classified the provided "Term" into one of the eight listed categories (the ninth being 'I don't know). The overall accuracy obtained for these tasks was *0.93*.
– *Language:* This crowdsourcing experiment is a language verification task for five languages. There are *25* tasks where the worker is shown text and the language and has to verify whether the

text is in the specified language. There are three options to choose from: *'Correct'*, *'Incorrect'* and *'I don't know'*. The overall accuracy obtained for these tasks was *0.86*.

In MTurk, we used the feature of assigning custom *"Qualification Types"* to the workers. This captures the notion of differences among workers and that non-answered questions fall into the hard tasks on the aggregate. After running an experiment, one can create a custom qualification type (e.g. 'best workers) and assign each worker to this type with a score ranging from 0 to 100. Then, when a new experiment is launched, one can use this qualification type as a criterion in the *"Worker Requirements"*.

We first performed a grid search in CrowdED using the fixed values from the experiments, as reported in Table 8, to determine the recommended values for each variable. For the total number of tasks, the grid search was from *10-200* and *10-100* for the *MetaCrowd* and Language experiments, respectively. For the proportion of tasks to train, the grid search was set from *0%* to *99%,* and the number of workers was set in the range of [*3, 5, 7, 9, 11, 13, 15, 17, 19]* for both experiments. There were *3078* and 3420 simulations performed, which took 2 hours and 14 minutes for the *MetaCrowd* and Language experiments, respectively.

Dataset	Total no. of workers	Proportion of tasks to train	No. of workers per task	Accuracy 2 stages
MetaCrowd	120.76 (SD 49.03)	0.324 (SD 0.147)	10.611 (SD 5.112)	0.968 (SD 0.015)
Language	66.886 (SD 22.783)	0.474 (SD 0.216)	9.634 (SD 4.994)	0.941 (SD 0.029)

Table 8. CrowdED recommended values for the real-world experiments

Based on the recommended values, we executed the two experiments in MTurk. For the *MetaCrowd* experiment, in Stage 1, an accuracy of *79%* was achieved for *525* (*32%* of total tasks) tasks with 10 workers per task and a total of *201* workers. For the Language experiment, in Stage 1, an accuracy of *96%* was achieved for *11* (*47%* of total tasks) tasks with *9* workers per task and a total of *21* workers. The performance of each worker was calculated as the total number of tasks that the worker answered correctly (concerning the gold standard) divided by the total number of tasks that the worker did. We then calculated the median from all the workers' performances, which was the threshold. All workers above this threshold were assigned the 'best workers' qualification and were chosen for Stage 2. The thresholds were *0.84* and 1 for the *MetaCrowd* and Language experiments, respectively. Based on this threshold, *94* and *16* workers were chosen for Stage 2. Workers who only performed one task and scored above were not selected for Stage 2.

We then executed Stage 2 with the remaining tasks and only chose the workers assigned the 'best workers' qualifications. We achieved an accuracy of *84%* and *98%* for *MetaCrowd* and Language tasks in Stage 2, which was higher than that of Stage 1. Interestingly, for the *MetaCrowd*

experiment, the overall accuracy was lower than the original real-world experiment in Stage 1. However, in comparison, the accuracy in Stage 2 was significantly higher than in Stage 1 in the re-run. After the evaluation experiments in Stage 2, the overall accuracy was lower than the real-world experiment, which we argue with the fact that the total cost was lower than the original experiment. For the Language experiment, the overall accuracy in Stage 2 was higher than in Stage 1 and significantly higher than in the real-world experiment. The cost was slightly higher than the original experiment, and we argue that the accuracy was significantly higher in the re-run. Table 9 reports the values for the real-world experiment results for the 2 Stages based on the CrowdED recommended values.

Dataset	no. of tasks	workers per task	workers	Stage 1 accuracy	Stage 1 cost	best workers	Stage 2 accuracy	Stage 2 cost	Total cost
MetaCrowd	1643	10	201	79%	$262.50	94	86%	167.85	430.35
Language	25	9	21	96%	$4.95	16	98%	3.6	8.55

***Table 9.** Results for the real-world experiments for the 2 Stages based on CrowdED recommended values*

4. RELATED WORK

Several empirical studies have determined the 'optimal' number of workers per task based on the quality of workers (Carvalho et al., 2016) or by studying different scenarios of the increasing complexity of tasks concerning worker quality (Sheng et al., 2008). Several models (Iren & Bilgen, 2014), (Dai et al., 2013), (Gao & Parameswaran, 2014), (Goel et al., 2017) provide approaches for cost-quality and cost-time optimization, respectively. Another strategy employed active learning algorithms (changing the assignments per task in real-time) to minimize the number of questions asked to the crowd to maximize the number of tasks (Mozafari et al., 2014). On the other hand, (Ul Hassan et al., 2013) and (Gadiraju et al., 2017) proposed approaches of self-rating by workers in combination with using gold-standard tasks for estimating the expertise of workers. However, the self-assessment does not always ensure high accuracy on the actual tasks. However, in these studies, the estimated optimal number is defined solely in terms of expected errors in the aggregated output; it is assumed that all workers are of the same quality; and are extremely expensive to adapt in a real-world experiment.

In (Ho & Vaughan, 2012), the authors propose a *Dual-Task Assigner* algorithm, which estimates unknown worker skill levels and assigns heterogeneous tasks to online arriving workers based on the estimation. In (Chen et al., 2013), the authors formulate the budget allocation problem in crowdsourcing into a *Markov Decision Process* and characterize the optimal policy using dynamic programming. Recently, the *CrowdTruth* methodology (Dumitrache et al., 2016) has been developed consisting of quality metrics for evaluating the example (input) data, crowd annotators

and their resulting annotations, which aims to provide valuable insights about the task design, annotation clarity, or annotator quality. However, these methods are applicable either during or after the task. Additionally, these studies either require that the pay be set based on the progress of the total number of tasks or that the number of workers is high, which can get expensive on a large scale.

MTurk and Figure Eight typically recommend reserving 10 - 30% of their tasks for "gold" test questions, whose answers are known and then dismissing workers who fail a disproportionate percentage of these tasks. Our results correspond to these recommendations. However (Bragg et al., 2016) argue that this policy may waste valuable budget and instead propose a model that (i) tests workers to determine their accuracy and (ii) getting work performed by good workers formulated as a partially-observable Markov decision process (POMDP). Then reinforcement learning over the POMDP is applied to dynamically improve the given base policy with experience. Another study (Fan et al., 2015) proposed *iCrowd* for on-the-fly estimation of the accuracies of a worker by evaluating her performance on the completed tasks and predicting which tasks the worker is well acquainted with. Finally, (Kobren et al., 2015) showed that setting goals dynamically, in conjunction with a reasonable allocation of tasks, increases the amount of information collected by the crowdsourcing system by up to 249%. However, the main drawback of these methodologies is that they can be expensive when implemented in the real world. CrowdED is distinct from all these studies as it offers a two-staged statistical model that can apriori, via simulations, estimate the number of workers assigned per task to gain maximum accuracy.

5. CONCLUSION

In this paper, we described a two-staged statistical guideline, CrowdED, for designing crowdsourcing tasks to estimate optimal workers and tasks' assignment apriori to obtain maximum accuracy on all tasks. CrowdED allows a requester to apriori simulates the 'optimal' values for each variable, thus reducing costs. CrowdED is open source, implemented in Python and can be executed in Jupyter notebooks. Furthermore, the 4 modules and the CrowdED function can be executed independently. We evaluated our proposed method by re-executing two real-world experiments using CrowdED recommended values and showing that we achieved higher accuracy and reduced costs compared to a random selection of values. Furthermore, we show that a two-stage methodology in crowdsourcing experiments leads to better overall experiment accuracy.

The current limitation of CrowdED is that it only simulates experiments with multiple-choice questions where the task is to choose one out of n valid answers. The method will not work for subjective questions or descriptive tasks where correct answers can be highly variable. We have used ten real-world experiments to define the apriori probabilities to feed the method. Within the scope of this paper, we do not compare it with other methods since a crowdsourcer usually has randomness as a baseline in a real scenario. Nevertheless, we posit that realistic synthetic data generation can tackle these issues in practice.

In future work, we will account for the optimisation algorithm's budgetary constraints and various parameters (e.g., hard tasks) and perform further simulated and real-world evaluations. Furthermore, we aim to extend CrowdED further to be able to simulate and recommend other types of crowdsourcing tasks (e.g. multiple answers in MCQs). Also, we will implement an interface such that a user can vary parameters and assumptions.

6. REFERENCES

Bragg, J., Mausam, & Weld, D. S. (2016). Optimal Testing for Crowd Workers. *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*.

Carvalho, A., Dimitrov, S., & Larson, K. (2016). How many crowdsourced workers should a requester hire? *Annals of Mathematics and Artificial Intelligence*, *78*(1), 45–72. https://doi.org/10.1007/s10472-015-9492-4

Chen, X., Lin, Q., & Zhou, D. (2013). Optimistic Knowledge Gradient Policy for Optimal Budget Allocation in Crowdsourcing. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th International Conference on Machine Learning* (Vol. 28, pp. 64–72). PMLR.

crowdflower. (2019). *Company categorizations (with URLs)*. https://data.world/crowdflower/company-categorizations

Dai, P., Lin, C. H., Mausam, & Weld, D. S. (2013). POMDP-based Control of Workflows for Crowdsourcing. *Artif. Intell.*, *202*(1), 52–85. https://doi.org/10.1016/j.artint.2013.06.002

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the 11th International AAAI Conference on Web and Social Media*, 512–515.

Dumitrache, A., Inel, O., Timmermans, B., Aroyo, L., Sips, R.-J., & Welty, C. (2016). CrowdTruth: Methodology for Gathering Annotated Data by Harnessing Disagreement in Crowdsourcing. *ICT Open*.

Fan, J., Li, G., Ooi, B. C., Tan, K., & Feng, J. (2015). iCrowd: An Adaptive Crowdsourcing Framework. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1015–1030. https://doi.org/10.1145/2723372.2750550

Fox, J.-P. (2008). Beta-binomial ANOVA for multivariate randomized response data. *British Journal of Mathematical and Statistical Psychology*, *61*(2), 453–470. https://doi.org/10.1348/000711007X226040

Gadiraju, U., Fetahu, B., Kawase, R., Siehndel, P., & Dietze, S. (2017). Using Worker Self-Assessments for Competence-Based Pre-Selection in Crowdsourcing Microtasks. *ACM Trans. Comput.-Hum. Interact.*, *24*(4), 30:1-30:26. https://doi.org/10.1145/3119930

Gao, Y., & Parameswaran, A. (2014). Finish Them!: Pricing Algorithms for Human Computation. *Proc. VLDB Endow.*, *7*(14), 1965–1976. https://doi.org/10.14778/2733085.2733101

Goel, K., Rajpal, S., & Mausam. (2017). Octopus: A Framework for Cost-Quality-Time Optimization in Crowdsourcing. *CoRR*, *abs/1702.03488*. http://arxiv.org/abs/1702.03488

Goto, S., Ishida, T., & Lin, D. (2016). Understanding Crowdsourcing Workflow: Modeling and Optimizing Iterative and Parallel Processes. *HCOMP*.

Haghighi, S., Jasemi, M., Hessabi, S., & Zolanvari, A. (2018). PyCM: Multiclass confusion matrix library in Python. *Journal of Open Source Software*, *3*(25), 729. https://doi.org/10.21105/joss.00729

Hassan, U. ul, Zaveri, A., Marx, E., Curry, E., & Lehmann", J. (2016). ACRyLIQ: Leveraging DBpedia for Adaptive Crowdsourcing in Linked Data Quality Assessment. *Knowledge Engineering and Knowledge Management*, 681–696.

Ho, C.-J., & Vaughan, J. W. (2012). Online Task Assignment in Crowdsourcing Markets. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 45–51.

Iren, D., & Bilgen, S. (2014). Cost of Quality in Crowdsourcing. *Human Computation*, *1*. https://doi.org/10.15346/hc.v1i2.14

KIT. (2019a). *DBpedia quality assessment- datatypes*. http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/experiments.html

KIT. (2019b). *DBpedia quality assessment—Interlinks*. http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/experiments.html

KIT. (2019c). *DBpedia quality assessment—Object values*. http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/experiments.html

Kobren, A., Tan, C. H., Ipeirotis, P. G., & Gabrilovich, E. (2015). Getting More for Less: Optimized Crowdsourcing with Dynamic Tasks and Goals. *WWW*.

Loni, B., Cheung, L., Riegler, M., Bozzon, A., Gottlieb, L., & Larson, M. (2014). Fashion 10000: An enriched social image dataset for fashion and clothing. *Proceedings of the 5th ACM Multimedia Systems Conference, MMSys 2014*, 41–46. https://doi.org/10.1145/2557642.2563675

Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., & Madden, S. (2014). Scaling up crowd-sourcing to very large datasets: A case for active learning. *Proceedings of the VLDB Endowment*, *8*(2), 125–136.

SHAKIL, M. (2009). Using Beta-binomial Distribution in Analyzing Some Multiple-Choice Questions of the Final Exam of a Math Course, and its Application in Predicting the Performance of Future Students*. *POLYGON - A Web-Based, Multi-Disciplinary Publication, Miami-Dade College, Hialeah Campus*.

Sheng, V. S., Provost, F., & Ipeirotis, P. G. (2008). Get another label? Improving data quality and data mining using multiple, noisy labelers. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 614–622.

Ul Hassan, U., O`Riain, S., & Curry, E. (2013). Effects of expertise assessment on the quality of task routing in human computation. *Proceedings of the 2nd International Workshop on Social Media for Crowdsourcing and Human Computation*.

VENANZI, M., Teacy, W., Rogers, A., & Jennings, N. (2015a). *Sentiment popularity—Amazon Mechanical Turk dataset*. University of Southampton. https://eprints.soton.ac.uk/376544/

VENANZI, M., Teacy, W., Rogers, A., & Jennings, N. (2015b). *Weather Sentiment—Amazon Mechanical Turk dataset*. University of Southampton. https://eprints.soton.ac.uk/376543/

Wikipedia. (2019). *Wikipedia image categorization*. https://d1p17r2m4rzlbo.cloudfront.net/wp-content/uploads/2016/03/Wiki-gender-image-categorization-DFE.csv

Yilmaz, E., Aslam, J. A., & Robertson, S. (2008). A New Rank Correlation Coefficient for Information Retrieval. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 587–594. https://doi.org/10.1145/1390334.1390435

Zaveri, A., & Dumontier, M. (2017). MetaCrowd: Crowdsourcing gene expression metadata quality assessment. *Bio-Ontologies*.

Zaveri, A., Hernandez Serrano, P., Desai, M., & Dumontier, M. (2018). CrowdED: Guideline for Optimal Crowdsourcing Experimental Design. *Companion Proceedings of the The Web Conference 2018*, 1109–1116. https://doi.org/10.1145/3184558.3191543