How Many Workers to Ask? Adaptive Exploration for Collecting High Quality Labels

Ittai Abraham [*] Omar Alonso Vasilis Kandylas Rajesh Patel Steven Shelford Aleksandrs Slivkins [†]

ABSTRACT

Crowdsourcing has been part of the IR toolbox as a cheap and fast mechanism to obtain labels for system development and evaluation. Successful deployment of crowdsourcing at scale involves adjusting many variables, a very important one being the number of workers needed per human intelligence task (HIT). We consider the crowdsourcing task of learning the answer to simple multiple-choice HITs, which are representative of many relevance experiments. In order to provide statistically significant results, one often needs to ask multiple workers to answer the same HIT. A stopping rule is an algorithm that, given a HIT, decides for any given set of worker answers to stop and output an answer or iterate and ask one more worker. In contrast to other solutions that try to estimate worker performance and answer at the same time, our approach assumes the historical performance of a worker is known and tries to estimate the HIT difficulty and answer at the same time. The difficulty of the HIT decides how much weight to give to each worker's answer. In this paper we investigate how to devise better stopping rules given workers' performance quality scores. We suggest adaptive exploration as a promising approach for scalable and automatic creation of ground truth. We conduct a data analysis on an industrial crowdsourcing platform, and use the observations from this analysis to design new stopping rules that use the workers' quality scores in a non-trivial manner. We then perform a number of experiments using real-world datasets and simulated data, showing that our algorithm performs better than other approaches.

Keywords: Crowdsourcing; label quality; ground truth; assessments; adaptive algorithms; multi-armed bandits.

1. INTRODUCTION

Crowdsourcing has become a central tool for improving the quality of search engines and many other large scale on-line services that require high quality assessments or labels. In this usage of crowdsourcing, a task or parts thereof are broadcast to multiple independent, relatively inexpensive workers, and their answers are aggregated. Automation and optimization of this process at a large scale allows to significantly reduce the costs associated with setting up, running, and analyzing experiments that contain such tasks.

In a typical industrial scenario that we consider in this paper, a *requester* has a collection of HITs, where each HIT has a specific, simple structure and involves only a small amount of work. We focus on multiple-choice HITs, that is, a HIT that contains a question with several possible answers. The goal of the requester is to learn the preference of the crowd on each of the HITs. For example, if a HIT asks whether a particular URL should be labeled as spam and most workers believe it should, then the requester would like to learn this. This abstract scenario with multiple-choice HITs covers important industrial applications such as relevance assessment and other optimizations for a web search engine and construction of training sets for machine learning algorithms. Obtaining high quality labels is not only important for both model training and development but also for quality evaluation.

The requester has two goals: extract high-quality information from the crowd (i.e., reduce the error rate), and minimize costs (e.g., in terms of money and time spent). There is a tension between these two goals; we will refer to it as the *quality-cost trade-off*. In practice, it is assumed that there is some noise from the crowd, so the requester defines in advance how many workers are needed per assignment for the whole task. This approach may not always be the right thing to do. For example, assessing the relevance of the query-URL pair (`facebook`, `www.facebook.com`) should need no more than one or two workers for such popular destination. In contrast, the pair (`solar storms`, `solarstorms.org`) would require more workers as the topic may not be familiar to some. Using a fixed number of workers may result in wasting resources for cases that are not needed or in not pooling more answers in assignments that require more power. Wouldn't it be useful if there is a flexible mechanism for adjusting the number of workers?

For cost-efficiency, one needs to take into account the heterogeneity in task difficulty and worker skills: some tasks are harder than others, and some workers are more skilled than others. Further, workers' relative skill level may vary from one task to another. In general, it is desirable to (1) use less aggregation for easier tasks, (2) use more skilled workers. The crowdsourcing system initially has a very limited knowledge of task difficulty, and possibly also of worker skills, but both can, in principle, be learned over time.

A common application that stems from the assessment scenario is the generation of ground truth or gold standard, usually called *gold* HITs or *honey pots*. These gold HITs are a set of HITs where the associated answers are known in advance. They can be a very effective mechanism to measure the performance of workers and data quality. Gold HITs are usually generated manually, typically

[*]VMware. Email: `ittaia@gmail.com`.
[†]All: Microsoft. {`omalonso, vakandyl, rajeshpa, steven.shelford, slivkins`}`@microsoft.com`.

SIGIR '16, July 17 - 21, 2016, Pisa, Italy
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4069-4/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2911451.2911514

by hired domain experts. This approach is not scalable: it is expensive, time consuming and error prone. We believe that much more automated systems should be available, whereby a requester starts with a relatively small gold HIT set for bootstrapping, and uses the crowd to generate arbitrarily larger gold HIT sets of high quality. A central challenge in designing a mechanism for automated gold HIT creation is cost-efficient quality control. With error-prone workers, one needs to aggregate the answers of several workers to obtain a statistically robust answer for a gold HIT.

We make the following contributions: (1) data analysis of HITs from a production platform, (2) design of two new stopping rule algorithms and (3) automatic generation of ground truth at scale. We now describe the specifics insights and improvements.

1. Data analysis of a crowdsourcing platform. We collected and analyzed a real-world data set from logs of UHRS, a large in-house crowdsourcing platform operated by a comercial search engine. We note that our data set cannot be easily replicated on a publicly accessible crowdsourcing platform such as Amazon Mechanical Turk. Indeed, this is a much larger data set (250,000 total answers) than one could realistically collect via targeted experiments (i.e., without access to platform's logs) because of budget and time limitations. Moreover, using realistic HITs in an open experiment tends to be difficult because of trade secrets.

Analysing the data, we make two empirical observations. First, we find that the difficulty of a random HIT is distributed near-uniformly across a wide range. Second, we investigate the interplay between HIT difficulty and worker quality, and we find that the high-quality workers are significantly better than the low-quality workers for the relatively harder tasks, whereas there is very little difference between all workers for the relatively easy tasks. These observations motivate our algorithms and allow us to construct realistic simulated workloads.

The above observations are based on a large-scale data analysis, which makes them valuable even if they may seem intuitive to one's common sense (albeit perhaps counterintuitive to someone else's). UHRS, Amazon Mechanical Turk, CrowdFlower, and others have similar architectural characteristics (e.g., HITs, task templates, payment system, etc.) so our data should be comparable to other platforms. Due to the proprietary nature of UHRS, this is the best information that we can share with the community.

2. Adaptive stopping rule algorithms. We consider obtaining a high-quality answer for a single HIT. We investigate a natural *adaptive* approach in which the platform adaptively decides how many workers to use before stopping and choosing the answer. The core algorithmic question here is how to design a *stopping rule*: an algorithm that at each round decides whether to stop or to ask one more worker. An obvious quality-cost trade-off is that using more workers naturally increases both costs and quality. In view of our empirical observation, we do not optimize for a particular difficulty level, but instead design *robust* algorithms that provide a competitive cost-quality trade-off for the entire range of difficulty.

As a baseline, we consider a scenario where workers are "anonymous", in the sense that the stopping rule cannot tell them apart. We design and analyze a simple stopping rule algorithm for this scenario, and optimize its parameters.

As workers vary in skill and expertise, one can assign quality scores to workers based on their past performance (typically, as measured on gold HITs). We investigate how these quality scores can help in building better stopping rules. While an obvious approach is to assign a fixed "voting weight" to each worker depending on the quality score, we find that more nuanced approaches perform even better. Given our empirical observations, we would like to utilize all workers for easy tasks, while giving more weight to better workers on harder tasks. As the task difficulty is not known a priori, we use the stopping time as a proxy: we start out believing that the task is easy, and change the belief in the "harder" direction over time as we ask more workers. We design a new adaptive stopping rule algorithm optimized for this setting. We conduct simulations based on the real workload, and conclude that this approach performs better than the "fixed-weight" approach.

We focus on the workers' quality scores that are given externally. This is for a practical reason: it is extremely difficult to design the entire crowdsourcing platform as a single algorithm that controls everything. Instead, one is typically forced to design the system in a modular way. In particular, while different requesters may want to have their own stopping rules, the crowdsourcing system may have a separate module that maintains workers' quality scores over different requesters.

3. Scalable gold HIT creation. Creating gold HITs presents additional challenges compared to the normal HITs. As the quality of the entire application (or successful experiment) hinges on the correctness of gold HITs, it is feasible and in fact desirable to route gold HITs to more reliable workers on the crowdsourcing platform. Worker quality is typically estimated via performance on the gold HITs that are already present in the system, so the estimates may be very imprecise initially, and gradually improve over time as more gold HITs are added. To find answers for individual HITs in a cost-efficient manner, one can use stopping rules as described above.

We tackle these challenges using ideas from *multi-armed bandits*, a problem space focused on sequentially choosing between a fixed and known set of alternatives with a goal to increase the cumulative reward and/or converge on the best alternative. A multi-armed bandit algorithm needs to trade off *exploration*, trying out various alternatives in order to gather information probably at the expense of short-term gains, and *exploitation*, choosing alternatives that perform well based on the information collected so far.

We consider a stylized model in which HITs arrive one by one, and the system sequentially assigns workers to a given HIT until it concludes that the answer is known with sufficient confidence. In particular, such system needs to "explore" the available workers in order to estimate their quality. We incorporate an insight from multi-armed bandits called *adaptive exploration*: not only the exploitation decisions, but also the exploration schedule itself can be adapted to the data points collected so far (e.g., we can give up early on low-performing alternatives). To implement adaptive exploration, we take a well-known approach from prior work on multi-armed bandits and tailor it to our setting, connecting it with the stopping rules described above. Our algorithm performs significantly better than the baseline uniform assignment of workers.

For algorithm evaluation we used the UHRS dataset discussed above, and also two previously published data sets from [26, 16]. Since the UHRS dataset is somewhat sensitive, we have been required to sanitize our results, and in particular we only evaluate on simulated data parameterized by the key properties of that dataset. One advantage of using a simulated workload is that one can replicate our algorithm evaluation (after choosing some values for the first column in Table 1). Also, we have been able to generate as much simulated data as needed for the experiments, whereas the available number of workers in the original data set was insufficient for some HITs.

The paper is organized as follows. Section 2 summarizes the related work on this area. We describe preliminary background in Section 3. We provide an analysis using data from an industrial crowdsourcing platform in Section 4. The design of a stopping rule for anonymous workers and its evaluation are described in Section 5. Similarly, the case for non-anonymous workers is described in Section 6. The gold HIT creation method is described in Section 7. Finally, conclusions and future work are outlined in Section 8.

2. RELATED WORK

The use of crowdsourcing as a cheap, fast and reliable mechanism for gathering labels was demonstrated in the areas of natural language processing [26], machine translation [8] and information retrieval [2] by running HITs on Amazon Mechanical Turk or CrowdFlower and comparing the results against an existing ground truth. While early publications have shown that majority voting is a reasonable approach to achieve good results, new strategies have emerged in the last few years. Jointly with that, several papers consider *task allocation*, the problem of allocating tasks to workers.

Oleson et al. [21] propose to use the notion of *programmatic gold*, a technique that employs manual spot checking and detection of bad work, in order to reduce the amount of manual work. Ground truth creation is a problem for new evaluation campaigns when no gold standard is available. Blanco et al. [6] rely on manual creation of gold answers for monitoring worker quality in a semantic search task. Scholer et al. [23] study the feasibility of using duplicate documents as ground truth in test collections.

Sheng et al. [24] design an algorithm that adaptively decides how many labels to use on a given HIT based on the distribution of all previously encountered HITs. Crucially, they assume that all HITs have the same difficulty for a given worker. However, our empirical evidence shows that HITs have widely varying difficulty levels; our algorithms are tailored to deal with this heterogeneity. Also, they optimize the quality of an overall classification objective, rather than the error rate.

Other approaches use the EM algorithm to estimate the workers' accuracy and the final HIT result at the same time [12]. The work presented in [16] is another algorithm based on EM, with several improvements. EM-based solutions use information from all the HITs in the data set and assume that a worker is answering many (or all) of these HITs and with more or less similar performance across them. Our approach is to consider each HIT individually and without using information from previously answered HITs. Because of this, we do not need to make the assumption that all the HITs are of similar difficulty. Additionally, it is not necessary to have the same workers answer multiple HITs. In fact, each HIT could be answered by a completely new set of workers. In a later section of this paper we make the additional assumption that we have knowledge of the overall quality of the workers, but we still consider that HITs could have varying (and unknown) difficulties.

Vox Populi [13] is a data cleaning algorithm that prunes low quality workers with the goal of improving a training set. The technique uses the aggregate label as an approximate ground truth and eliminates the workers that tend to provide incorrect answers.

Karger et al. [19] optimize task allocation given budgets. Unlike ours, their solution is non-adaptive, in the sense that the task allocation is not adapted to the answers received so far. Further, [19] assume known Bayesian prior on both tasks and judges, whereas we do not.

From a methodology perspective, CrowdSynth [18] focuses on addressing consensus tasks by leveraging supervised learning.

Parameswaran et al. [22] consider a setting similar to our stopping rules for HITs with two possible answers. Unlike us, they assume that all HITs have the same difficulty level, and that the (two-sided) error probabilities are known to the algorithm. They focus designing algorithms for computing an optimal stopping rule.

Settings similar to stopping rules for anonymous workers, but incomparable on a technical level, were considered in prior work, e.g. [4], [17], [5], [11], [20], [1].

For scalable gold HIT creation, our model emphasizes explore-exploit trade-off, and as such is related to multi-armed bandits; see [9, 7] for background on bandits and [25] for a discussion of explore-exploit problems that arise in crowdsourcing markets. Our algorithm builds on a bandit algorithm from Auer et al. [3].

Ho et al. [15], Abraham et al. [1] and Chen et al. [10] consider models for adaptive task assignment with heterogeneity in task difficulty levels and worker skill that are technically different from ours. In [15], the interaction protocol is "inverted": workers arrive one by one, and the algorithm sequentially and adaptively assigns tasks to each worker before irrevocably moving on to the next one. The exploration schedule in [15] is non-adaptive, unlike ours, in the sense that it does not depend on the observations already collected. The solution in [1] focuses on a single HIT. The algorithm in [10] develops an approach based on Bayesian bandits that requires exact knowledge of the experimentation budget and the Bayesian priors.

3. PRELIMINARIES

A HIT is a question with a set of S of possible answers. For each HIT we assume that there exists one answer which is the correct answer (more on this below under "probabilistic assumptions"). A requester has a collection of HITs, which we call a *workload*. The goal of the requester is to learn what is the correct answer for each HIT. The requester has access to a crowdsourcing system.

We model a stylized crowdsourcing system that operates in rounds. In each round the crowdsourcing systems chooses one HIT from the workload and a worker arrives, receives the HIT, submits her answer and gets paid a fixed amount for her work. The crowdsourcing system needs to output an answer for each HIT in the workload. The algorithm can adaptively decide for each HIT how many workers to ask for answers. We mostly focus on a single HIT. In each round, a worker arrives and submits an answer to this HIT. The algorithm needs to decide whether to stop (stopping rule) and if so, which answer to choose (selection rule).

There are two measures to be minimized in such an algorithm: (1) the *error rate* for this workload (the percentage of HITs for which the algorithm outputs the wrong answer), and (2) the *average cost* for this workload (the average cost per HIT paid to the workers by the algorithm). Formally this is a bi-criteria optimization problem. If all workers are paid equally, the average cost is simply the average number of rounds.

Probabilistic Assumptions. We model each worker's answer as a random variable over S, and assume that these random variables are mutually independent. We assume that the most probable answer is the same for each worker. For the purposes of this paper, the "correct answer" is just the most probable answer, and this is the answer that we strive to learn.The difference between the probability of the most probable answer and the second most probably answer is called the *bias* of a given worker on a given HIT. This quantity, averaged over all workers, is the *bias* of a given HIT. A large bias (close to 1) corresponds to our intuition that the HIT is very easy: the error rate is very small, whereas a small bias (close to 0) implies that the HIT is hard, in the sense that it is very difficult to distinguish between the two most probable options. Here, our notion of easy/hard HITs is objective (reflecting agreement with majority), rather than subjective (reflecting workers' sentiments). Hereafter we use the bias of a HIT as a measure of its hardness. In particular, we say that HIT A is *harder* than HIT B if the bias of A is smaller than the bias of B.

4. DATA ANALYSIS

We performed our analysis using data from UHRS, a large in-house crowdsourcing platform operated by Microsoft. UHRS is used by many different internal groups for evaluation, label collection, and machine learning applications. The tasks range from TREC-like evaluations to domain specific labeling and experimentation. In particular, UHRS is used to gather training and evaluation data for various aspects of the search engine.

Using the logs of UHRS, we collected a data set from a variety of tasks and workers. In that data set, we selected all tasks that contained at least 50 HITs, and all HITs with at least 50 answers. These HITs have been used for training and/or quality control, which explains an unusually large number of answers per HIT. This large number has been essential for our purposes. We considered all HITs in all these tasks. This gave us a data set containing 20 tasks, 3,000 workers, 2,700 HITs, and 250,000 total answers. For each HIT we computed the majority answer, which we considered as the "correct" answer. Details of the different types of HITs, design templates, and other specific metrics are left out due to proprietary information.

Empirical Biases of HITs. Workers' replies to a given HIT are, at first approximation, IID samples from some fixed distribution $\mathcal{D}$. A crucial property of $\mathcal{D}$ is the difference between the top two probabilities, which we call bias of this HIT; note that the bias completely defines $\mathcal{D}$ if there are only two answers. Informally, larger bias corresponds to easier HIT. We study the distribution over biases in our workload. For each HIT, we consider the empirical frequencies of answers, and define "empirical bias" as the difference between the top two frequencies. We plot the CDF for empirical biases in Figure 1.

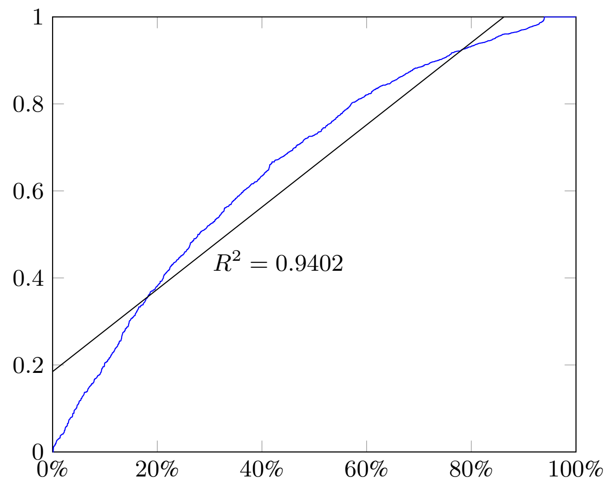

Figure 1: CDF for the empirical bias of HITs.

We conclude that HITs have a wide range of biases: some are significantly more difficult than others. In particular, tailoring a decision rule to HITs with a specific narrow range of biases is impractical. Further, we observe that the empirical distribution is, roughly, near-uniform. We use this observation to generate the simulated workload in the next section.

Error Rates. For each worker, we compute the average error rate across all HITs that she answered. According to that, we split all workers into 9 equally sized groups, from best-performing (W_0) to worst-performing (W_8). Similarly, for each HIT we compute the average error rate across all workers that answered it. We split all HITs into 9 equally-sized groups, from easiest (H_0) to most difficult (H_8). Let $\texttt{error}(W_i, H_j)$ be the average error rate of the workers in the worker group W_i when answering the HITs in the HIT group H_j.

To make our main finding clearer, and also because our data set is somewhat sensitive, we report a 9-by-8 table (see Table 1): for each HIT group $H_i, i = 0 \dots 8$ and each worker group $W_j, j = 1 \dots 8$, the corresponding cell contains the difference

$$\texttt{error}(W_i, H_j) - \texttt{error}(W_0, H_j). \quad (1)$$

The table is also visualized as a heat map in Figure 2.

%	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
H_0	0	0	0	1	1	1	2	4
H_1	1	1	2	2	3	4	6	15
H_2	1	3	3	4	6	8	11	20
H_3	1	4	4	7	7	11	16	27
H_4	4	7	8	12	13	17	23	36
H_5	5	9	11	14	18	20	26	43
H_6	7	11	15	18	22	25	30	47
H_7	11	14	19	21	25	26	33	48
H_8	19	24	27	29	31	35	39	50

Table 1: Error rates for different worker/HIT groups. The cell (W_i, H_j) contains the difference (1), in percent points.

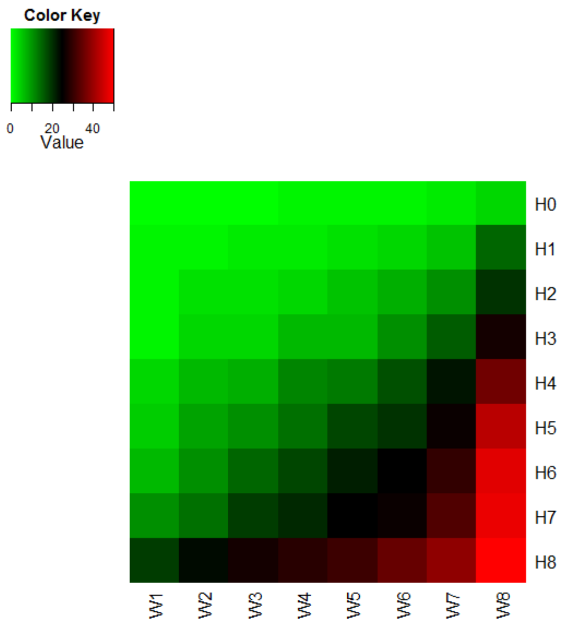

Figure 2: Error rates for different worker/HIT groups.

Findings. From Table 1, we make the following observations. For difficult tasks ($H_6 \dots H_8$) the set of good judges ($W_0 \dots W_2$) is significantly better (has a lower error rate) than the set of bad judges ($W_6 \dots W_8$). For easy tasks ($H_0 \dots H_2$) there is very little difference between *all judges* (expect perhaps for the very worst judges).

These observations are robust to changing the number of HIT and worker groups (from 5 to 9). To summarize, *the difference in performance between good and bad workers is much more significant for harder HITs than for easier HITs.* Accordingly, we devise algorithms that tend to use all workers for easier HITs, and favor better workers for more difficult HITs.

5. STOPPING RULE FOR ANONYMOUS WORKERS

We start with a simpler case when workers are anonymous, in the sense that there is no prior information on which workers are better than others. Absent such information, we treat all workers equally: essentially, we give each worker's vote the same weight.

5.1 Algorithm

For simplicity, let us assume there are only two answers for a HIT: A and B. In each round t, let $V_{A,t}$ and $V_{B,t}$ be the number of workers that vote for A and B, respectively. Note that $t = V_{A,t} + V_{B_t}$. Our stopping rule is as follows:

$$\text{Stop if } |V_{A,t} - V_{B,t}| \geq C\sqrt{t} - \epsilon t. \tag{2}$$

Here $\epsilon \geq 0$ and $C \geq 0$ are parameters that need to be chosen in advance. After the algorithm stops, the selection rule is simply to select the most frequent answer. Note that the right-hand side is not an integer, so we can randomly round it to one of the two closest integers in a way that is proportional to the fractional part.

Discussion. Our intuition is that each worker's reply is drawn IID from some fixed distribution over answers; recall that the bias of a HIT is the difference between the top two probabilities in this distribution. For two answers:

$$\texttt{bias} = |\Pr[A] - \Pr[B]|$$

Informally, the meaning of parameter ϵ is that we are willing to tolerate a higher error rate for HITs with $\texttt{bias} \leq \epsilon$, in order to improve the error-cost trade-off for the entire workload. We find in our simulations that a small value of ϵ performs better than $\epsilon = 0$.

Parameter C controls the error-cost trade-off: increasing it increases the average cost and decreases the error rate. In practice, the parameters (C, ϵ) should be adjusted to typical workloads to obtain the desirable error-cost trade-off.

Analysis. For the sake of analysis, let us consider a slight modification of algorithm (2) in which parameter C is proportional to $\log t$ (we view this dependence as minor compared to the $\sqrt{t}$ term).

We prove that our algorithm returns a correct answer with high probability if $\texttt{bias} \geq \epsilon$. We consider two hypotheses:

(H1) The correct answer is A and $\texttt{bias} \geq \epsilon$,

(H2) The correct answer is B and $\texttt{bias} \geq \epsilon$.

Effectively, if one hypothesis is right, our algorithm rejects the other with high probability.

With $\epsilon = 0$, the expected cost (stopping time) is on the order of $\texttt{bias}^{-2}$, in line with standard results on biased coin tossing. Using $\epsilon > 0$ relaxes this to $(\epsilon + \texttt{bias})^{-2}$.

Lemma 5.1 *Fix $\delta \in (0,1)$. Consider the algorithm (2) with parameters $\epsilon > 0$ and $C = C_t = \sqrt{\log(t^2/\delta)}$. Suppose this algorithm is applied to a HIT with $\texttt{bias} = \epsilon_0$.*

(a) *If $\epsilon_0 \geq \epsilon$ then the algorithm returns a correct answer with probability at least $1 - O(\delta)$.*

(b) *The expected cost (stopping time) is at most $O\left(\rho^{-2} \log \frac{1}{\delta\rho}\right)$, where $\rho = \epsilon + \epsilon_0$.*

PROOF. W.l.o.g., suppose $\Pr[A] \geq \Pr[B]$. Consider the difference $Z_t = V_{A,t} - V_{B,t} - \epsilon_0 t$, where t ranges over rounds. The increments $Z_t - Z_{t-1}$ are independent random variables with mean 0 and values ± 1, so Z_t is a random walk. Therefore for each t:

$$\Pr[|Z_t| \leq C_t\sqrt{t}] \geq 1 - O(\delta/t^2). \tag{3}$$

by a standard application of the *Azuma-Hoeffding Inequality*. Taking the Union Bound over all t, it follows that

$$\Pr\left[|Z_t| \leq C_t\sqrt{t} \quad \text{for all } t\right] \geq 1 - O(\delta). \tag{4}$$

For part (a) assume that hypothesis (H1) holds, i.e. that $\epsilon_0 \geq \epsilon$, but the algorithm returns an incorrect answer, i.e. stops at some round t so that answer B is chosen. We show this cannot happen if the high-probability event in (4) holds. Indeed, at such round t:

$$V_{B,t} - V_{A,t} > C_t\sqrt{t} - \epsilon t$$
$$Z_t < (\epsilon - \epsilon_0)t - C_t\sqrt{t} < -C_t\sqrt{t}.$$

The latter contradicts the high-probability event in (4).

For part (b), let T be the stopping time. Consider round t such that $t \geq (2C_t/\rho)^2$. For any such round, the high-probability event $\{Z_t > -C_t\sqrt{t}\}$ implies that

$$V_{A,t} - V_{B,t} > -C_t\sqrt{t} + \epsilon_0 t \geq C_t\sqrt{t} - \epsilon t,$$

so the algorithm stops at round t or earlier, i.e., $T \leq t$. By (3), we conclude that $\Pr[T > t] < O(\delta/t^2)$. Now, there exists $t_0 = O(\rho^{-2} \log \frac{1}{\delta\rho})$ such that $t \geq (2C_t/\rho)^2$ for all $t \geq t_0$. Therefore:

$$\mathbb{E}[T] = \textstyle\sum_{t=1}^{\infty} \Pr[T > t] \leq t_0 + \sum_{t>t_0} \delta/t^2 = t_0 + O(\delta).$$

So the expected stopping time $\mathbb{E}[T]$ is as small as we claimed. □

Extension to multiple answers. One can extend the stopping rule (2) to more than two answers in an obvious way. At time t, let $A^*(t)$ and $B^*(t)$ be the answers with the largest and second-largest number of votes, respectively. The stopping rule is

$$\text{Stop if } V_{A^*(t),t} - V_{B^*(t),t} \geq C\sqrt{t} - \epsilon t. \tag{5}$$

The selection rule is to select the most frequent answer.

Lemma 5.1 easily carries over to multiple answers. (The proof considers a separate random walk for each pair of answers A, B:

$$Z_t^{A,B} = V_{A,t} - V_{B,t} - t(\Pr[A] - \Pr[B]),$$

obtains high-probability event $\{|Z_t^{A,B}| \leq C_t\sqrt{t}\}$ as in (3), and then conditions on the intersection of all such events.)

5.2 Experimental Results

We used a simulated workload, consisting of 100,000 HITs, each with two answers. For each HIT, the bias towards the correct answer (the difference between the probabilities of the two answers) was chosen uniformly at random in the interval $[0.1, 0.6]$. This closely matches an empirical distribution of biases, as we have found in the previous experiments. For each worker answering this HIT, the answer was chosen independently at random with the corresponding bias.

For each pair (ϵ, C) of parameters, running our algorithm on a single HIT gives a two-fold outcome: the cost and whether the correct answer was chosen. Thus, running our algorithm on all HITs in our workload results in two numbers: average cost and error rate (over all HITs). We plot these pairs of numbers on a coordinate plane where the axes are average cost and error rate. Thus, fixing ϵ and varying C we obtain a curve on this plane, which we call the *varying-C curve*.

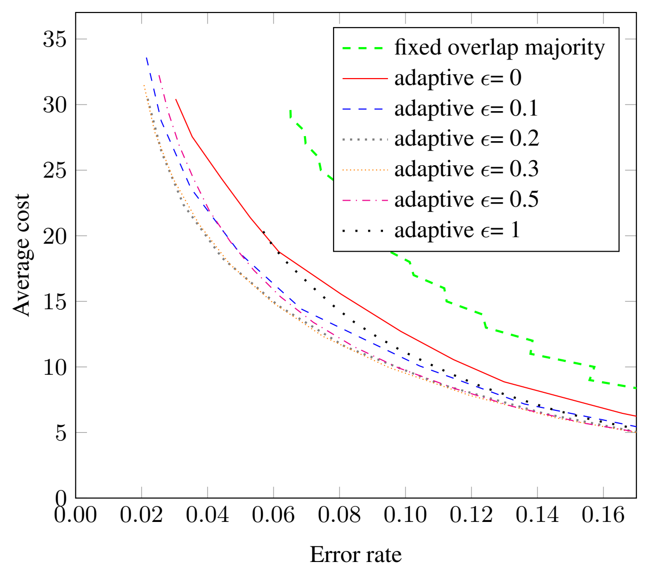

Figure 3: Cost-quality trade-off for fixed overlap majority and the adaptive algorithm on a simulated data set.

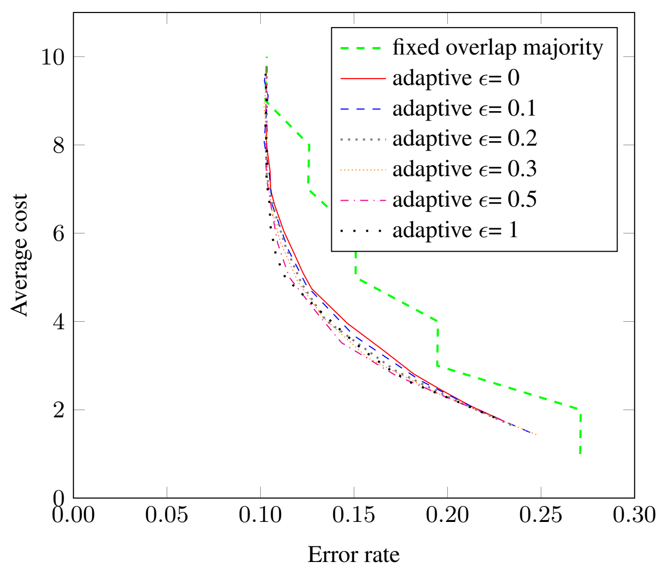

Figure 4: Cost-quality trade-off for fixed overlap majority and the adaptive algorithm on the RTE data set.

We consider several values for ϵ, ranging from 0 to 1. For each value of ϵ, we plot the corresponding varying-C curve (Figure 3). We also plot, as baseline, the fixed overlap majority algorithm, which uses a fixed number of annotations per HIT (we vary from 1 to 30) and uses simple majority voting (breaking ties randomly). This technique is used in [26]. Surprisingly, we find that, up to some minor noise, for any two varying-C curves it holds that one lies below another. This did not have to be the case, as two curves could criss-cross. If one varying-C curve lies below another varying-C curve, this means that the ϵ parameter for the former curve is always better: for any C, it gives better average cost for the same error rate. Thus, we find that for any two significantly different values of parameter ϵ, one value is better than another, regardless of the C. From Figure 3, we find that the most promising range for ϵ is $[.2, .3]$. We have omitted the less interesting values for clarity.

We repeat the same experiment with the NLP RTE data set from [26]. The RTE data set contains 800 HITs and 10 annotations per HIT. For the fixed overlap majority algorithm we randomly sampled a fixed number of worker labels per HIT (varying the fixed number to produce the curve), whereas for the other algorithms we randomly permuted the order of the worker labels. The results we report are the averages of the error rates and costs over 100 runs (Figure 4). As with the simulated data set, the adaptive algorithm performs better than the fixed overlap majority. Because of the 10 annotations per HIT, it is impossible to do better than the minimum error rate of about 0.11, which is achieved when all available annotations are used. However, the adaptive algorithm can achieve approximately the same error with much lower average cost (about 6 instead of 10).

Finally, we repeat the same experiment with the adult data set from [16]. This data set contains classifications of web pages into four categories (G, PG, R, X), depending on the adult content on the page. There are 500 web pages with approximately 100 labels per page. We were unable to obtain from the authors the original gold labels of the web pages that they used for their experiments, therefore we computed the majority answer per web page and treated that as gold. Using the same adult data set, we also run the EM-based algorithm from [16], known as Get Another Label (GAL). The original algorithm also contains a majority voting step that can be used to break ties using label priors (we call GAL with this step GAL majority vote). For GAL, GAL majority vote and the fixed overlap majority algorithms we randomly sampled a fixed number of worker labels per HIT (varying the fixed number to produce the curves), whereas for the other algorithms we randomly permuted the order of the worker labels. The results we report are the averages of the error rates and costs over 100 runs (Figure 5). We have omitted some curves for the adaptive algorithm for clarity. The performance of the adaptive algorithm is better than fixed overlap majority and GAL. This is not too surprising, considering that both GAL and fixed overlap majority use a fixed number of labels for every HIT so they cannot really decrease the cost by stopping early like adaptive does. The GAL majority vote algorithm performs comparably to the fixed overlap majority which is expected as they only differ on how they break ties. Somewhat surprisingly, GAL performed worse than the GAL majority vote. We believe this was caused by the way we generated the ground truth. Since we could not obtain the original ground truth of the adult data set, we used the majority answer as ground truth and this may have impacted the results.

6. STOPPING RULE FOR NON-ANONYMOUS WORKERS

Depending of the task and qualifications required, some workers may be better than others, and one can often estimate who is better by looking at the past performance. We assume workers have a one-dimensional personal measure of *expertise* or skill level, which influences their error rate on HITs. Further, we assume we have access to a *reputation system* which can (approximately and coarsely) rank workers by their expertise level. We develop a weighted version of the stopping rule from Section 5 that is geared to take advantage of such a reputation system. We begin by describing a general weighted stopping rule, then detail how we use it.

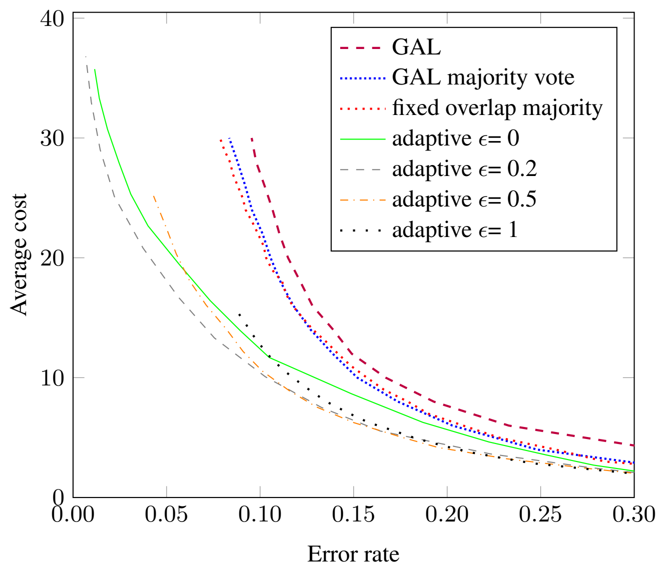

Figure 5: Cost-quality trade-off for fixed overlap majority, Get Another Label (GAL) and the adaptive algorithm on the adult data used by Get Another Label.

6.1 Algorithm

In each round t, the worker is assigned weight w_t. In general, the weights may depend on the available information about the worker and the task. Also, the stopping rule can update the next worker's weight depending on the number t itself. For now, we do not specify *how* the weights are assigned. Absent any prior information on the workers, all weights are 1. Such stopping rules will be called *unweighted*; we have discussed them in Section 5.

Fix some round t. The weighted vote $V_{A,t}$ for a given answer A is defined as the total weight of all workers that arrived up to (and including) round t and chose answer A. For simplicity, assume there are only two answers: A and B. Our stopping rule is as follows:

$$\text{Stop if } |V_{A,t} - V_{B,t}| \geq C\sqrt{\textstyle\sum_{s=1}^{t} w_s^2} - \epsilon \textstyle\sum_{s=1}^{t} w_s. \quad (6)$$

Here $C > 0$ and $\epsilon \in [0, 1)$ are parameters that need to be chosen in advance. Note that in the unweighted case ($w_t \equiv 1$), this reduces to Equation (2). Our default selection rule is to choose the answer with the largest weighted vote. We call this the *deterministic* selection rule.

Discussion. The goal for weighted stopping rule is identical to the unweighted case: among the two hypotheses (H1) and (H2), reject the one that is wrong.

Letting $W_{t,q} = (\sum_{s=1}^{t} w_s^q)^{1/q}$, we can re-write the stopping rule (6) more compactly as

$$\text{Stop if } |V_{A,t} - V_{B,t}| \geq C\, W_{t,2} - \epsilon W_{t,1}. \quad (7)$$

The meaning of the right-hand side is as follows. $Z_t = V_{A,t} - V_{B,t}$ can be viewed as a biased random walk: its increments $Z_t - Z_{t-1}$ are independent random variables with values $\pm w_t$ and mean $\epsilon_0 = \Pr[A] - \Pr[B]$. The expected drift of this random walk is $\mathbb{E}[Z_t] = \epsilon_0\, W_{t,1}$. Thus, the term $\epsilon W_{t,1}$ in (7) is a lower bound on the expected drift assuming either (H1) or (H2) holds. The meaning of the $C\, W_{t,2}$ term in (7) is that $W_{t,2}$ is the best available upper bound on the standard deviation of Z_t.

Extension to multiple answers. It is easy to extend the stopping rule (6) to more than two answers. Let A and B be the answers with the largest and second-largest weighted vote, respectively. The stopping rule is

$$\text{Stop if } V_{A,t} - V_{B,t} \geq C\, W_{t,2} - \epsilon W_{t,1}. \quad (8)$$

Defining the weights. We restrict our attention to *coarse* quality scores. This is because a reputation system is likely to be imprecise in practice, especially in relation to a specific HIT. So more fine-grained quality scores, especially continuous ones, are not likely to be meaningful.

Suppose each worker is assigned a coarse quality score qty, e.g. $\texttt{qty} \in \{\texttt{good}, \texttt{average}, \texttt{bad}\}$. Our general approach, which we call *reputation-dependent exponentiation*, is as follows. For each possible quality score qty we have an initial weight $\lambda_{\texttt{qty}}$ and the multiplier $\gamma_{\texttt{qty}}$. If in round t a worker with quality score qty is asked, then her weight is

$$w_t = \lambda_{\texttt{qty}}\, \gamma_{\texttt{qty}}^{t-1}$$

A notable special case is *time-invariant weights*: $\gamma_{\texttt{qty}} = 1$.

The intuition is that we want to gradually increase the weight of the better workers, and gradually decrease the weight of the worse workers. The gradual increase/decrease may be desirable because of the following heuristic argument. As we found empirically (see Table 1), the difference in performance between good and bad workers is more significant for hard HITs, whereas for very easy HITs all workers tend to perform equally well. Therefore we want to make the difference in *weights* between the good and bad workers to be more significant for harder HITs. While we do not know a priori how difficult a given HIT is, we can estimate its difficulty as we get more answers. One very easy estimate is the number of answers so far: if we asked many workers and still did not stop, this indicates that the HIT is probably hard. Thus, we increase/decrease weights gradually over time.

6.2 Experimental Results

Simulated workload. We use the real data 9-by-9 table of error rates for different worker and HIT groups to generate a simulated workload that consists of 100,000 HITs, all with two answers, and 100 workers that answer all these HITs. We split workers uniformly across worker groups, and split HITs uniformly among HIT groups. For each worker and each HIT, the correct answer is chosen with the probability given by the corresponding cell in the table. We define a coarse quality score depending on the worker group: the best three worker groups were designated *good*, the middle three *average* and the last three *bad*. This quality score is given as input to the algorithm.

Algorithms tested. We tested several "reputation-dependent exponentiation" algorithms. Recall that the weights in each such algorithm are defined by the initial weights $\lambda_{\texttt{qty}}$ and the multipliers $\gamma_{\texttt{qty}}$ for each quality score $\texttt{qty} \in \{\texttt{good}, \texttt{average}, \texttt{bad}\}$. For convenience, we denote the initial weights $\vec{\lambda} = (\lambda_{\texttt{good}}, \lambda_{\texttt{average}}, \lambda_{\texttt{bad}})$ and likewise the multipliers $\vec{\gamma} = (\gamma_{\texttt{good}}, \gamma_{\texttt{average}}, \gamma_{\texttt{bad}})$. We experimented with many assignments for $(\vec{\lambda}, \vec{\gamma})$. Below we report on several paradigmatic versions:

1. No weights (all weights are set to 1, $\vec{\lambda} = (1, 1, 1)$):
   - *Fixed overlap majority:* Same as section 5.2, uses a fixed number of annotations (i.e. overlap) per HIT and computes the simple majority answer. Ties are broken randomly. We vary the overlap parameter to produce the cost-error curve.

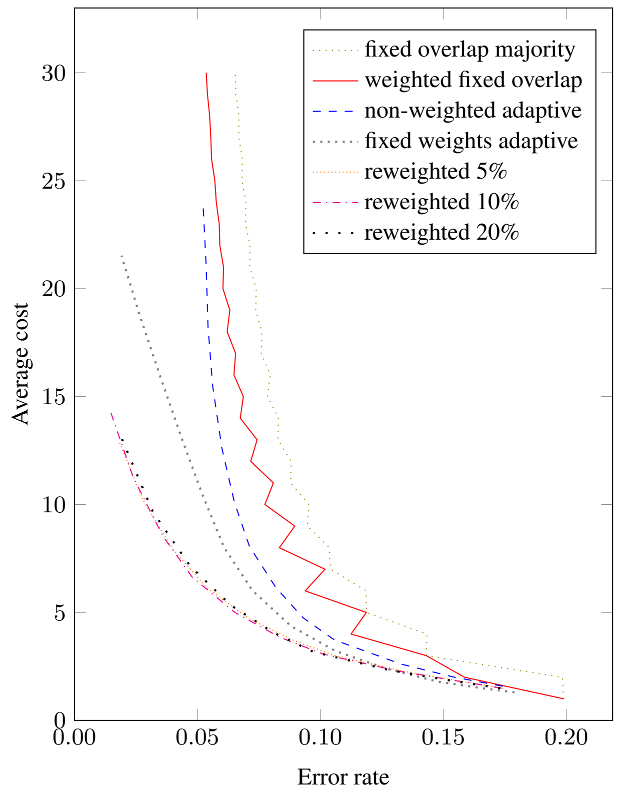

Figure 6: Cost-error trade-off for weighted stopping rules. (For all varying-C curves, $\epsilon = 0.2$.)

- *Non-weighted adaptive:* The adaptive algorithm from the previous section, which assumes all the workers are anonymous.

2. Time-invariant weights (multipliers are set to 1, $\vec{\gamma} = (1, 1, 1)$):

   - *Weighted fixed overlap:* Same as the fixed overlap majority algorithm, but the majority is weighted, using weights $\vec{\lambda} = (1.2, 1, 0.8)$.
   - *Fixed weights adaptive:* Similar to the adaptive algorithm, but uses the weighted stopping rule with weights $\vec{\lambda} = (1.2, 1, 0.8)$. The weights do not change during execution.

3. Time-varying weights. All weights start equal to 1 ($\vec{\lambda} = (1, 1, 1)$), but they increase/decrease by certain multipliers per round. We consider different multipliers $\vec{\gamma}$, so that weights change as follows.

   - *Reweighted adaptive 5%:* The weights change by 5% per round with multipliers $\vec{\gamma} = (1.05, 1, 0.95)$.
   - *Reweighted adaptive 10%:* The weights change by 10% per round with multipliers $\vec{\gamma} = (1.1, 1, 0.9)$.
   - *Reweighted adaptive 20%:* The weights change by 20% per round with multipliers $\vec{\gamma} = (1.2, 1, 0.8)$.

For each assignment of $(\vec{\lambda}, \vec{\gamma})$, we consider several values for the parameter ϵ, and for each ϵ we plotted a varying-C curve in the error rate vs. expected cost plane. To showcase our findings, some representative choices are shown in Figure 6.

Our findings. As in the previous section, we find that (up to some minor noise) for any two varying-C curves, one lies below another. This enables comparisons between different algorithms that are valid for all choices of parameter C. We conclude:

- Using weights is better than not. The weighted fixed overlap algorithm performs better than the non-weighted version. Similarly, the adaptive algorithm performs better with weights (fixed or time-varying) than in the non-weighted (anonymous workers) case.
- The adaptive algorithm performs better than the fixed overlap majority, even if the adaptive does not use weights, and the fixed overlap does.
- For the adaptive, it is better to update the weights per round, rather than keeping them fixed.
- How much the weights get updated does not have a big effect in performance (for the range of 5% to 20% that was tested).
- It is better to use $\epsilon > 0$. The best value for ϵ is usually in the range $[.2, .3]$, and the effect of changing ϵ within this range is usually very small. This follows from experiments not shown in figure 6, but also supported in the experiments of the previous section and Figure 3.

We also experimented with various other combinations of weight updating schemes and multiplier values, which are not shown in the figure. We tried updating the weights every four rounds rather than every round (updating by $\gamma^4_{\texttt{qty}}$, accordingly, for each quality score $\texttt{qty}$), and we found that updating every round performs better. We also tried updating only the weights of the good workers (or only the weights of the bad workers) and the differences were very small.

Further, we investigated the effect of the magnitude of the multipliers $\vec{\gamma}$. We tried the previously mentioned weighted adaptive algorithm with multipliers that modify the worker weights by 5%, 10%, 20%, 30%, 40% and 50% (for example, $\vec{\gamma} = (1.3, 1, 0.7)$ for 30% weight updates). We found the differences to be very small, with the updates of 5% and 10% to be very slightly better.

7. SCALABLE GOLD HIT CREATION

We turn to the problem of scalable gold HIT creation, as described in the Introduction. We consider a stylized model with heterogeneity in worker quality but not in HIT difficulty. The system processes a stream of HITs, possibly in parallel. Each HIT is assigned to workers, sequentially and adaptively, at unit cost per worker, until the gold HIT answer is generated with sufficient confidence or the system gives up. Worker skill levels are initially not known to the algorithm, but can be estimated over time based on past performance. The goal is to minimize the total cost while ensuring low error rate.

Algorithm. We adopt the following idea from prior work on multi-armed bandits: for each worker, combine exploration and exploitation in a single numerical score, updated over time, and at each decision point choose a worker with the highest current score [27, 14, 3]. This score, traditionally called an *index*, takes into account both the average skill observed so far (to promote exploitation) and the uncertainty from insufficient sampling (to promote exploration). Over time, the algorithm zooms in on more skilled workers.

We use a simple algorithm which builds on [3, 1]. For each worker i, let t_i be the number of performed HITs for which the algorithm has generated a gold HIT answer, and let t_i^+ be the number of those HITs where the worker's answer coincides with the gold HIT. If $t_i \geq 1$, we define this worker's index as

$$\texttt{Index}_i = \frac{t_i^+}{t_i} + \frac{1}{\sqrt{t_i}}.$$

Note that $\texttt{Index}_i \leq 2$. For initialization, we set $\texttt{Index}_i = 2$.

Now that we've defined $\texttt{Index}_i$, the algorithm is very simple:

- At each time step, pick a worker with the highest index, breaking ties arbitrarily.
- For each HIT, use the unweighted stopping rule (5) to decide whether to stop processing this HIT. Then the gold HIT answer is defined as the majority answer.

Experimental setup. To study the empirical performance of our index-based algorithm, we use a simulation parameterized by real data as follows. We focus on HITs with binary answers. We have $1,000$ workers and each worker generates a correct answer for each HIT independently, with some fixed probability (*success rate*) which reflects her skill level. The success rate of each worker is drawn independently from a realistic "quality distribution" $\mathcal{D}_{\texttt{qty}}$.

We determined $\mathcal{D}_{\texttt{qty}}$ by examining a large set ($> 1,500$) of real workers from our internal platform (cf. Section 4), and computing their average success rates over several months. Thus we obtained an empirical quality distribution, which we approximate by a low degree polynomial (see Figure 7).

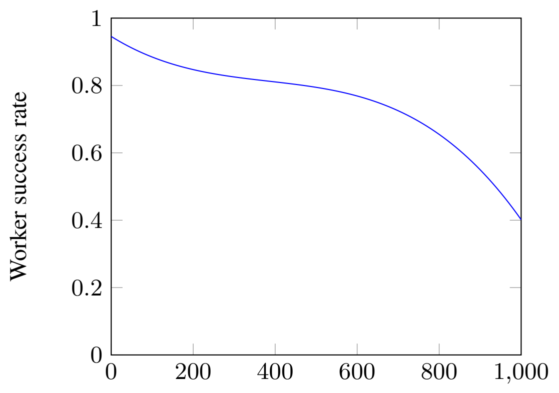

Figure 7: Worker quality distribution $\mathcal{D}_{\texttt{qty}}$.

We compare our index-based algorithm to a naive algorithm, called `Random`, which assigns each HIT to a random worker. Both algorithms use the same unweighted stopping rule (5). In our simulation, each algorithm processes HITs one by one (but in practice the HITs could be processed in parallel).

Recall that the stopping rule comes with two parameters, ϵ and C. We consider three different values of ϵ, namely $\epsilon = 0$, $\epsilon = 0.05$ and $\epsilon = 0.1$. (Recall that according to our simulations in Section 5, $[0.05, 2]$ is the most promising range for ϵ.) For each algorithm and each value of ϵ, we vary the parameter C to obtain different cost vs. quality trade-offs. For each value of C, we compute 5K gold HITs using each algorithm. Thus, for each algorithm and each value of ϵ we obtain a varying-C curve. The simulation results are summarized in Figure 8. The main finding is that our index-based algorithm reduces the per-HIT average cost by 35% to 50%, compared to `Random` with the same error rate. Recall that the cost here refers to the number of workers, which in practical terms translates to both time and money. Thus, we suggest adaptive exploration, and particularly index-based algorithms, as a very promising approach for automated gold HIT creation.

8. DISCUSSION AND CONCLUSIONS

In this paper, we mainly focus on the issue of deciding how many workers to ask for a given HIT. The number of workers asked defines a trade-off between the cost of the HIT and the error rate of the final answer. We propose an adaptive stopping rule which, every time a worker is asked, decides whether to stop or continue asking another worker. The stopping rule takes into account the differences in the workers' answers and the uncertainty from the limited number of these answers. This allows asking few workers for easy HITs, where their answers are mostly identical, and thus incurring low cost. On the other hand, for harder HITs, more workers are asked in order to maintain a low error rate. A simpler scheme that uses a fixed number of workers per HIT wastes answers on the easy HITs and lacks enough answers on the harder HITs.

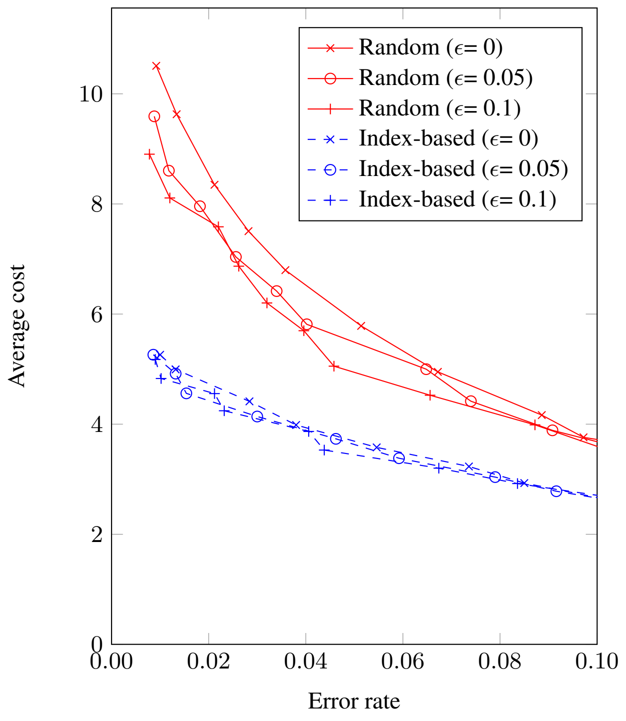

Figure 8: Simulation results.

If workers' skill levels are approximately known from their past performance, we can improve the stopping rule to take the skill levels into account. The difficulty of a new HIT is, as before, assumed to be unknown. From our data analysis we know that all workers tend to perform well on easy HITs, whereas on harder HITs the skill level of the workers tends to make a big difference. We can thus estimate the HIT difficulty by the number of answers when the stopping rule decided to stop. We use this information to reweight the answers of the workers according to their known skill, so that for harder HITs we rely more on the better workers. With other EM-based algorithms, the assumption is that we do not know much about the workers and we try to assign a score to them that corresponds to how good they are (e.g., spammers get a low score, whereas workers that are giving correct answers get a high score). The EM-based algorithms try to estimate the worker scores at the same time as estimating the HIT answers. If workers give answers that agree with the estimated ones, then they tend to get high scores. Our adaptive algorithm instead assumes that we know how good or bad the workers are. What it tries to estimate is how easy or hard the HIT is, and what should be the HIT answer. The idea is that for easy HITs, even poorly performing workers are quite reliable So, if we can figure out that a HIT is easy we can rely on pretty much every worker, whereas if a HIT is hard we should discount the answers of the bad workers. The adaptive algorithm estimates the difficulty of the HIT based on how much the workers agree or disagree and then assigns a weight to each worker's answer that relies on both the estimated difficulty of the HIT and the worker quality. This also means that the worker weights differ from HIT to HIT.

One can envision an approach where the worker skill is not known beforehand but can be learned algorithmically. For example, after the stopping rule decides to stop and produce a final answer for the HIT, we could compare the worker's answer to the final answer. If their answer matches, we can assume they gave a correct answer. This approach is particularly suitable to the problem of scalable gold HIT creation. However, further research is required to establish if this can produce accurate results in practice or if it leads to "self-fulfilling loops" where the workers who are considered skilled provide the same wrong answer. Such answer is then interpreted as the "correct" answer by the system, which in turn reinforces the belief that these workers are highly skilled.

While our stopping rules return a single answer for a given HIT, they can be extended to HITs with *several* correct answers. For example, if the vote difference is small between the top two answers, but large between the second and the third answer, then we could stop and output the top two answers as both being correct. With similarly simple modifications, the rules can be expanded to deal with HITs in which the answers correspond to specific numerical values. In that case, it is not only the vote difference that matters but also the difference between the corresponding numerical values. These extensions are the subject of future research.

Acknowledgments. We thank Jennifer Wortman Vaughan for providing valuable feedback.

9. REFERENCES

[1] Ittai Abraham, Omar Alonso, Vasilis Kandylas, and Aleksandrs Slivkins. Adaptive crowdsourcing algorithms for the bandit survey problem. In *26th COLT*, 2013.

[2] Omar Alonso and Stefano Mizzaro. Using crowdsourcing for TREC relevance assessment. *Inf. Process. Manage.*, 48(6):1053–1066, 2012.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[4] R. E. Bechhofer, S. Elmaghraby, and N. Morse. A single-sample multiple decision procedure for selecting the multinomial event which has the highest probability. *Annals of Mathematical Statistics*, 30:102–119, 1959.

[5] R. E. Bechhofer and D. Goldsman. Truncation of the bechhofer-kiefer-sobel sequential procedure for selecting the multinomial event which has the largest probability. *Communications in Statistics – Simulation and Computation*, B14:283–315, 1985.

[6] Roi Blanco, Harry Halpin, Daniel M. Herzig, Peter Mika, Jeffrey Pound, Henry S. Thompson, and Duc Thanh Tran. Repeatable and reliable search system evaluation using crowdsourcing. In *34th SIGIR*, 2011.

[7] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[8] Chris Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *EMNLP*, pages 286–295, 2009.

[9] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge Univ. Press, 2006.

[10] Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient for optimal budget allocation in crowdsourcing. In *30th ICML*, 2013.

[11] Paul Dagum, Richard M. Karp, Michael Luby, and Sheldon M. Ross. An optimal algorithm for monte carlo estimation. *SIAM J. on Computing*, 29(5):1484–1496, 2000.

[12] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[13] Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *22nd COLT*, 2009.

[14] J. C. Gittins. Bandit processes and dynamic allocation indices (with discussion). *J. Roy. Statist. Soc. Ser. B*, 41:148–177, 1979.

[15] Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *30th ICML*, 2013.

[16] Panagiotis Ipeirotis, Foster Provost, and Jing Wang. Quality Management on Amazon Mechanical Turk. In *HCOMP*, 2010.

[17] J. T. Ramey Jr. and K. Alam. A sequential procedure for selecting the most probable multinomial event. *Biometrica*, 66:171–173, 1979.

[18] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *11th AAMAS*, 2012.

[19] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *25th NIPS*, pages 1953–1961, 2011.

[20] Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. Empirical bernstein stopping. In *25th ICML*, pages 672–679, 2008.

[21] David Oleson, Alexander Sorokin, Greg Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *Human Computation Workshop*, 2011.

[22] Aditya G. Parameswaran, Hector Garcia-Molina, Hyunjung Park, Neoklis Polyzotis, Aditya Ramesh, and Jennifer Widom. Crowdscreen: algorithms for filtering data with humans. In *ACM SIGMOD*, pages 361–372, 2012.

[23] Falk Scholer, Andrew Turpin, and Mark Sanderson. Quantifying test collection quality based on the consistency of relevance judgements. In *Proceeding of the 34th International ACM SIGIR*, pages 1063–1072, 2011.

[24] Victor S. Sheng, Foster J. Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *14th KDD*, 2008.

[25] Aleksandrs Slivkins and Jennifer Wortman Vaughan. Online decision making in crowdsourcing markets: Theoretical challenges. *SIGecom Exchanges*, 12(2), December 2013.

[26] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.

[27] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.