Human Computation and Crowdsourcing: Works in Progress and Demonstration Abstracts
An Adjunct to the Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing

Using Worker Quality Scores to Improve Stopping Rules (Work in Progress)*

Ittai Abraham, Omar Alonso, Vasilis Kandylas,
Rajesh Patel, Steven Shelford, and Alex Slivkins
{ittaia, omalonso, vakandyl, rajeshpa, stevsh, slivkins}@microsoft.com
Microsoft

Abstract

We consider the crowdsourcing task of learning the answer to simple multiple-choice microtasks. In order to provide statistically significant results, one often needs to ask multiple workers to answer the same microtask. A stopping rule is an algorithm that for a given microtask decides for any given set of worker answers if the system should stop and output an answer or iterate and ask one more worker. A quality score for a worker is a score that reflects the historic performance of that worker. In this paper we investigate how to devise better stopping rules given such quality scores. We conduct a data analysis on a large-scale 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 conduct a simulation based on a real-world workload, showing that our algorithm performs better than the more naive approaches.

Introduction. A crowdsourcing system allows to quickly and cheaply obtain and aggregate information from a large online community of people. One of the most prominent uses of crowdsourcing is in learning the *wisdom of the crowd*. In a typical industrial scenario that we consider in this paper, a surveyor has a collection of microtasks (also called *HITs*: Human Intelligence Tasks), where each HIT has a specific, simple structure and involves only a small amount of work. We focus on multiple-choice HITs: each HIT consists of a question with several possible answers. The goal of the surveyor is to learn the majority opinion of the crowd on each HIT. This abstract scenario covers important industrial applications such as relevance assessment and other optimizations in a web search engine and construction of training sets for machine learning algorithms. For example, if a HIT asks whether a particular URL should be labeled as spam and most workers believe it should, then the surveyor would like to learn this.

The surveyor has two objectives: 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 fundamental tradeoff between these two objectives; on a high level, our goal is to optimize this tradeoff.

We focus on obtaining a high-quality answer for a single HIT. We investigate a natural *adaptive* approach: the platform adaptively decides how many workers to use before stopping and choosing the answer. The core algorithmic question here is to design a *stopping rule*: an algorithm that at each round decides whether to stop or to ask one more worker. An obvious tradeoff here is that using more workers naturally increases both costs and quality.

Workers vary in skill and expertise, and one can assign quality scores to workers based on their past performance. We investigate how these quality scores can help in building better stopping rules. While an obvious approach is to give more weight to workers with better quality scores, we find that more nuanced approaches perform even better.

We provide analysis of a real crowdsourcing system and design new stopping rule algorithms based on our conclusions. There could be other ways a quality score can be used, but we argue that there is good reason to consider systems that only use them to optimize the stopping rule. In particular, we assume that each worker gets paid the same micro-payment amount for each microtask she answers and we assume that the platform cannot choose which workers show up and answer the tasks. These assumptions match the current behaviour of many large scale crowdsouring platforms. Morover adding differential payments and/or differential task assignment introduces a wide range of complexities. We believe that modifying these assumptions may raise some interesting research questions that are beyond the scope of this paper.

Our contribution - data analysis. We consider a large collection of HITs from *UHRS* – a large in-house crowdsourcing platform operated by Microsoft. 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. We partition HITs (resp., workers) into groups according to the empirical HIT difficulty (resp., worker quality). We construct a coarse matrix of task difficulty vs workers' quality, where each cell contains the average error rate when workers in a given worker group answer HITs in a given HIT group. Looking at this matrix, we observe that the better-quality workers are significantly better than the low-quality workers for the relatively harder tasks. Moreover, there is very little difference

*The full paper is available on arxiv.org.

between all workers for the relatively easy tasks. We find this surprising, as we previously believed that better-quality workers are *always* much better than worse-quality workers. These observation allow us to construct realistic simulated workloads for our experiments.

Our contribution - algorithms. Since HIT difficulty is near-uniformly distributed across a rather large range, we do not optimize for a particular difficulty level, but instead design *robust* algorithms that provide a competitive cost-quality tradeoff for the entire range of difficulty. As a baseline, we consider a scenario where workers are "anonymous", in the sense that they have no history and therefore no quality scores. We design and analyze a simple stopping rule for this scenario, and optimize its parameters using a realistic simulated workload.

We then devise algorithms that use worker quality scores. Given the above empirical observations, we would like to utilize all workers for easy tasks, while giving more weight to better workers on harder tasks. As we do not know a priori whether a given HIT is easy or hard, we use the stopping time as a proxy for task difficulty. Informally, we start out believing that the task is easy, and change the belief in the "harder" direction over time as we ask more workers. Accordingly, we start with all workers having equal weights, and gradually modify the weights over time: increase weights for better workers and/or decrease weights for worse workers. We consider several algorithms based on this weight-modifying approach, and compare them to more obvious algorithms that do not change weights over time. We conduct simulation based on the real workload, and conclude that the weight-modifying approach performs better.

Our model. We assume that each HIT has a correct answer. A **surveyor** has a collection of HITs, called a **workload**, and strives to learn the correct answer for each task. The surveyor has access to a stylized *crowdsourcing system* that operates in rounds. In each round it chooses one HIT from the workload and a **worker** arrives, receives the task, submits his answer and gets paid a fixed amount for her work. The system needs to output an answer for each HIT.

We focus on designing stopping rules for a single HIT. In each round, a worker arrives and submits an answer to this HIT, and our algorithm decides whether to stop, and if so, which answer to output. For a given workload, there are two objectives to minimize: the **error rate** – the fraction of tasks for which the algorithm outputs the wrong answer, and the **average cost** per task paid to the workers by the algorithm.

We consider a plane whose coordinates are the error rate and the average cost. The performance of a given algorithm on a given workload corresponds to a point on this plane. Our algorithms have a parameter that controls the error rate vs. average cost tradeoff. So each *parameterized* algorithm corresponds to a curve on this plane.

For the analysis, and also for intuition, we model a given worker's answer to a given HIT as an independent random variable. Moreover, we assume that the most probable answer to each HIT is the same for all workers.

For a given HIT, we consider the difference in average probability between the two most probable answers, where the averages are taken over all workers. We treat this difference as an objective measure of difficulty of the HIT: a large difference (close to 1) implies the HIT is very easy, and a small difference (close to 0) implies that it is very hard.

Related work. Much work on crowdsourcing is usually done using platforms like *Amazon Mechanical Turk* or *CrowdFlower*. (Snow et al. 2008) found that on these platforms, the majority voting is a good approach to achieve quality. (Sheng, Provost, and Ipeirotis 2008) explore adaptive schemes with Bayesian assumptions. A study on machine translation quality uses preference voting for combining ranked judgments (Callison-Burch 2009). Budget-optimal task allocation (Karger, Oh, and Shah 2011) focuses on a non-adaptive solution to the task allocation problem given a prior distribution on both tasks and judges (while we focus adaptive solutions and do not assume priors on tasks). CrowdSynth addresses consensus tasks by leveraging supervised learning (Kamar, Hacker, and Horvitz 2012). A recent line of research is adding a crowdsourcing layer as part of a computation engine, e.g. (Franklin et al. 2011).

Settings similar to stopping rules for anonymous workers (but technically incomparable) were considered in prior work, e.g. (Ramey Jr and Alam 1979; Dagum et al. 2000; Mnih, Szepesvári, and Audibert 2008; Abraham et al. 2013).

References

Abraham, I.; Alonso, O.; Kandylas, V.; and Slivkins, A. 2013. Adaptive crowdsourcing algorithms for the bandit survey problem. In *26th COLT*.

Callison-Burch, C. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *EMNLP*.

Dagum, P.; Karp, R. M.; Luby, M.; and Ross, S. M. 2000. An optimal algorithm for monte carlo estimation. *SIAM J. on Computing* 29(5):1484–1496.

Franklin, M. J.; Kossmann, D.; Kraska, T.; Ramesh, S.; and Xin, R. 2011. Crowddb: answering queries with crowdsourcing.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *11th AAMAS*.

Karger, D. R.; Oh, S.; and Shah, D. 2011. Iterative learning for reliable crowdsourcing systems. In *25th NIPS*.

Mnih, V.; Szepesvári, C.; and Audibert, J.-Y. 2008. Empirical bernstein stopping. In *25th ICML*.

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

Sheng, V. S.; Provost, F. J.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *14th KDD*.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*.