Self-correcting Crowds

Walter S. Lasecki
University of Rochester
252 Elmwood Ave.
Rochester, NY 14627 USA
wlasecki@cs.rochester.edu

Jeffrey P. Bigham
University of Rochester
252 Elmwood Ave.
Rochester, NY 14627 USA
jbigham@cs.rochester.edu

Abstract
Much of the current work in crowdsourcing is focused on increasing the quality of responses. Quality issues are most often due to a small subset of low quality workers. The ability to distinguish between high and low quality workers would allow a wide range of error correction to be performed for such tasks. However, differentiating between these types is difficult when no measure of individual success is available. We propose it is possible to use higher quality workers to compensate for lower quality ones, without explicitly identifying them, by allowing them to observe and react to the input of the collective. In this paper, we present initial work on eliciting this behavior and discuss how it may be possible to leverage self-correction in the crowd for better performance on continuous real-time tasks.

Keywords
Crowdsourcing, Human Computation, Error Correction

ACM Classification Keywords
H.1.2 [User/Machine Systems]: Human information processing; H.4.1 [Office Automation]: Groupware

General Terms
Algorithms, Reliability, Performance, Human Factors

Introduction
Using the *crowd*, a dynamic group of workers available on-demand, has been proven to be very effective for completing tasks that automatic systems currently cannot. However, workers in the crowd vary greatly in ability and attentiveness. Most low quality worker are not malicious, but instead either don't understand the task, have impairments such as a poor connection, or are simply lazy. We propose the idea of *self-correcting crowds* created by giving high quality workers the ability to compensate for less effective ones.

Our approach currently focuses only on changing the information available to workers. This method can generalize to nearly any type of task, and is especially advantageous to those needing quick responses, using smaller crowds. We present findings from a set of initial tests, and discuss design considerations for eliciting self-correcting crowds in existing systems.

Premise
Crowdsourcing relies on workers contributing pieces of knowledge towards a task. Current methods focus on trying to extract the correct answer from each worker and either averaging responses, or using the majority selection. However, the final answer can be biased by the incorrect input of a small group of workers. This is especially true when large crowds are not available, as is often the case in real-time crowdsourcing.

Our goal is to let attentive workers who understand the task compensate for those who do not by *overshooting* their answer, to skew the average towards what they view to be the correct answer. A majority of low quality workers are not actively trying to compromise the final output, but are either lazy or don't understand the task. We propose that enabling workers to accurately skew the final answer will primarily be used by higher quality workers, and ignored by low quality workers.

Continuous real-time crowd sourcing systems such as Legion [2], which enables crowd control of user interfaces, are of particular interest. Giving workers the ability to make these to adjust for low quality answers at the same time they submit input, rather than using a post-processing stage, allows for much quicker corrections even in domains that lack easily computable measures of quality. Self-correction allows us to benefit from properties of high quality groups of workers, without needing to identify them.

Self-Correction
We define self-correction for tasks using two main methods of combining input used in crowdsourcing:

- **Averaging:** Tasks where worker inputs are combined to reach a final decision. Potential inputs must occur on a continuum, and the result of the average may not be the choice of any single worker.

- **Voting:** Tasks where answers cannot be combined, so a majority decision is most often used. The final answer must be the input of at least one worker.

For tasks that average responses, correction can be performed by encouraging workers to *overshoot* the correct answer – selecting instead an answer that causes the average of the crowd to come closer to the correct one. This type of adjustment is done in real-time and does not require any method of automatically determining worker quality. For real-time crowdsourcing tasks, this means increasing accuracy without adding additional response time.

For tasks that elicit votes for distinct responses, workers can be asked to rank their choices (Borda count), or otherwise use a voting system that has them rate multiple options. This can be seen as converting the task to a multi-variable averaging problem, where each option's score is being averaged. Combining the answers will result in a crowd ranking of options. Workers can vote their selection higher and reduce the average ranking of others they believe to be incorrect. The higher an incorrect answer appears in the crowd ranking, the lower a worker will rate it in order to reduce its likelihood of winning. While this will enable correction, it should be noted that this type of voting scheme also allows for final choices that are not the selection of any single worker. For instance, if half of workers rank three options in the order 1,2,3 while the other half rank them 3,2,1 then option 2 will win even though it was no ones top choice. For most tasks this does not detract from the reliability of the final answer. This type of voting is often considered more reliable, and as such is used by many large-scale institutions[1].

Although we focus on cases where large crowds are not available, the ability of self-correction to be applied in both of these types of task allows for most current crowdsourcing tasks to take advantage of this.

[1] For example, some sports leagues such as the NCAA use large crowds and have contributors provide rankings in this way

Initial Tests

To test self-correction, we implemented a simple web-based game in which workers try to navigate a cursor through a series of barriers, each containing small openings, by controlling the horizontal position of the cursor. Figure 1 shows the game presented to workers.

In order to compensate for other workers, it must be possible for high quality workers to view the status of the crowd decision. That way, they can understand the need for and effect of overshooting. Each worker is shown both the position of the crowd and the worker's individual cursor. The position of the crowd cursor was determined by averaging all of the worker positions. Using the left and right arrow keys, workers were able to move their cursor horizontally to find the best choice of position for both the crowd cursor and/or their own to make it through the opening in the next barrier.

Tests were run using workers from Amazon's Mechanical Turk service, with two different payment schemes. In the first, workers were paid one cent per barrier for each getting their own cursor through the opening, and two cents for each time the crowd's cursor made it through an opening. In the second, we did not pay workers unless the crowd cursor made it through. As a control, we ran one set of each test without the crowd position visible to workers and only rewarded workers based on their own cursor position.

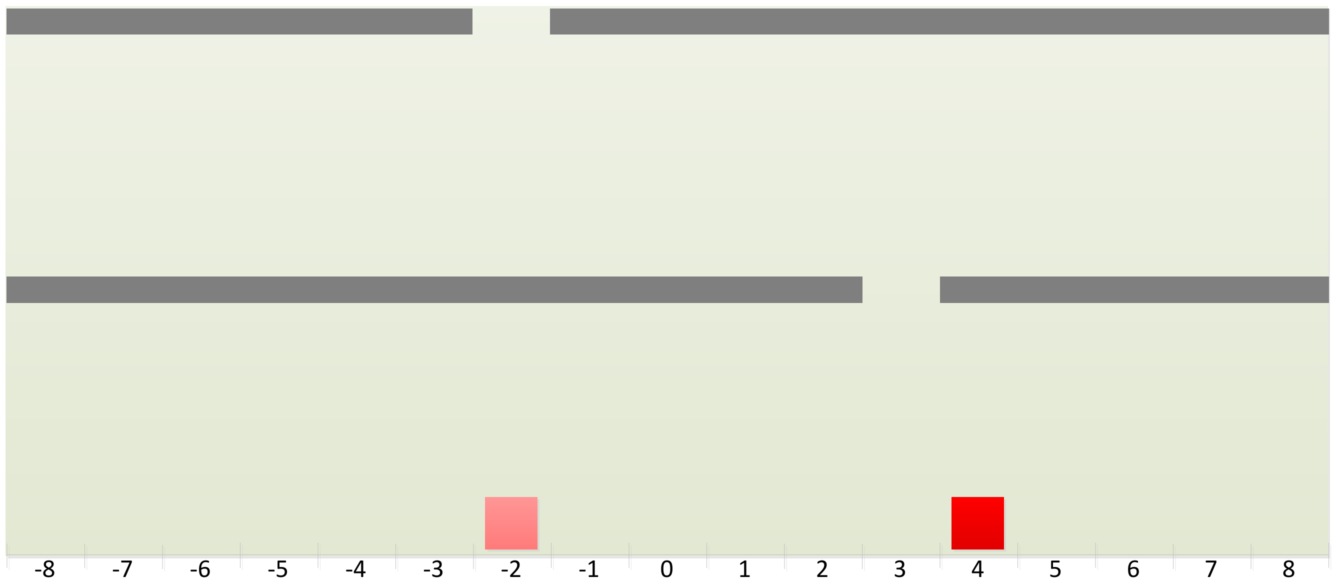

Figure 1. The test interface from the perspective of a single worker. The grey barriers move from the top of the screen to the bottom, at which point the cursors either collide (task fails), or pass through the opening (task succeeds). Workers were asked to control the lightly shaded cursor using arrow keys. The position of the crowd cursor (dark red block) was calculated by averaging the individual positions of all connected workers. The worker in the figure is trying to *overshoot* the correct answer (position 3) to pull the crowd cursor toward the opening.

Observations

Our initial results showed that Mechanical Turk workers often connected and remained mostly idle, making the overall improvement we saw not statistically significant.

We found that a subset of workers attempted to correct for the errors of others. However, the compensation needed to elicit this behavior was higher than that needed to simply recruit workers to control their own cursor. As such, workers participating in the first set of trials resorted to focusing on their own cursor. Workers in the second trial who first tried to overshoot abandoned the task quicker than those who did not. We believe this is due to a disconnect between the rewards given to users and their level of effort.

Besides the difference in motivating price, we also found that workers needed a clear understanding not just of how the system derived the crowd position but also of how their decision was affecting the crowd.

Future Work

Our initial work has shown that self-correcting behavior can be elicited in the crowd. It has also demonstrated that there are additional concerns for the design of the interface, beyond those of the original task.

Design

Designing interfaces that allow workers to see enough about other user's inputs to correct for them, but do not enable them to collude in a way that would undermine the motivation scheme of the system (such as agreeing on a single answer just to get paid) is difficult. Allowing users to see the current final response in a task using averaging enables users to not only correct, but also to agree on to the answer if identifying the answer is easier than solving the problem (which is not the case in our test). Also, when trying to adapt these methods to work with any task with no external metric, concerns arise that presenting the group decision reveals what will be considered the 'best' answer by the system.

Motivation

Our approach also introduces additional ambiguity into the system, since we discourage high quality workers from converging to the correct answer. However, it may be possible to use this to identify workers who are helping to correct the answer based on behavior. This would provide a means of rewarding workers based on quality, even though one was not initially available.

Future Tests

Workers in our tests found the results of making corrections to be too uncertain to make it worth it to commit to influencing the crowd cursor position over their own. This problem is similar to the Stag Hunt problem from game theory, in which players are guaranteed a small reward individually, but a larger one if they can all agree on an action. In order to fix this, future tests will both increase the reward for the crowd's success, and make it clearer what current influence a worker's current action is having on the crowd cursor, instead of only showing the aggregate position of the crowd as we do now. We will also focus on the case were workers are only paid if the crowd succeeds in passing through the barrier. This is also a better analogue to real tasks, where it may not be possible to identify when a single worker succeeds.

New Methods

It may be possible to enable stronger control of the crowd decision. For example, the range of the options limits the our method of skewing answers since the amount one worker can skew a decision is at most half the size of the total range. However, by artificially extending the range of options (i.e. symmetrically doubling the initial range), it may be possible to give workers the ability to move the crowd anywhere in the actual range. Final answers can be computed by mapping the new range back to the old.

Background

Previous work has explored using workers to correct for the errors made by others. The ESP Game [5] had workers agree on the content of an image before accepting a label as correct. We start with simultaneous job complete as a basis for self-correcting tasks.

Soylent [1] uses groups of workers to check the error finding and corrections performed by other groups. The find-fix-verify process can be seen as a non-real-time version of self-correction. It is important to note that to accomplish this, Soylent enforced a pattern to extract the behavior, and did not rely on the individual quality of workers. Our method aims to accomplish this by using self-selecting groups drawn from the same crowd by providing appropriate information and motivation.

Massively Multiplayer Pong [3] uses a similar control scheme to our test game. Players each control a “paddle” in a game of pong. Players are broken up into two teams, and the position of each team’s collective paddle is determined by the average position of all of the members of the team. Players are able to see the position of all players in the game, and the crowd as a whole. We expect that self-correction did take place in this setting, but no study of it was done performed.

Legion [2] is a system that enables continuous real-time control of existing interfaces. Legion uses *input mediators* to combine the input of multiple users in real-time. It has been used to control interfaces for a wide range of tasks, including robot navigation, word processing, support for predictive keyboards, and activity recognition. Currently, input is collected from workers and merged over very short time spans in order to simulate continuous control by a single user. There is also no external metric that can be used to determine if the current actions will lead to the correct result prior to the task ending. In the future, we will extend this continuous real-time platform to take advantage of self-correcting crowds.

Conclusion

We have presented idea of self-correcting crowds, and methods that use workers’ own ability to identify invalid input, before a final decision is reached, to correct mistakes. This can be used to improve the reliability of real-time crowdsourcing by compensating for the input of low quality workers even using small crowds, and without adding significant delay.

Depending on the type of task, we can either ask workers to directly compensate for others, or introduce voting systems that allow the same behavior. We have also discussed how future modifications to voting schemes, such as artificially increasing the range of choices, may lead to better control. Furthermore, it may be possible to use this new worker behavior to identify high quality workers in a crowd.

References

[1] Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., and Panovich, K. (2010). *Soylent: A Word Processor with a Crowd* Inside. In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2010). New York, NY. p313-322.

[2] Lasecki, W.S., Murray, K.I., White, S., Miller, R.C. and Bigham, J.P. (2011). *Real-time Crowd Control of Existing Interfaces*. In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST 2011). Santa Barbra, CA. p23-32.

[3] Massively multiplayer pong. (2006). http://collisiondetection.net

[4] Surowiecki, J. (2004). *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Little, Brown.

[5] von Ahn, L., Dabbish, L. (2004). *Labeling images with a computer game*. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2004). Vienna, Austria. p319-326.