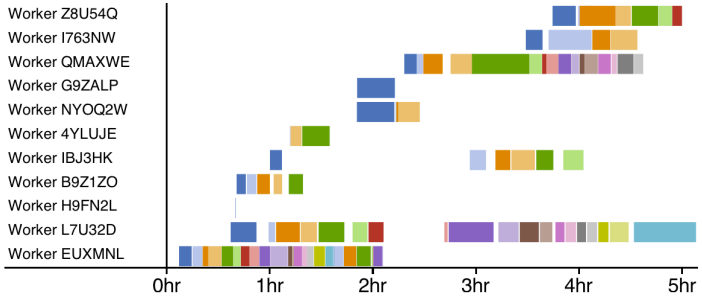
Shepherding the Crowd: Managing and Providing Feedback to Crowd Workers

Steven P. Dow
Brie Bunge
Truc Nguyen
Scott R. Klemmer
Stanford HCI Group
Stanford, CA 94305 USA
[spdow, bbunge, nguyen90, srk]
@stanford.edu

Anand Kulkarni[1]
Björn Hartmann[2]
[1]Industrial Engineering &
Operations Research
[2]Computer Science Division
University of California, Berkeley
Berkeley, CA 94720 USA
[anandk, bjoern] @
[ieor, eecs].berkeley.edu



Online crowd workers currently overlap, but have no interaction.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
Micro-task platforms provide a marketplace for hiring people to do short-term work for small payments. Requesters often struggle to obtain high-quality results, especially on content-creation tasks, because work cannot be easily verified and workers can move to other tasks without consequence. Such platforms provide little opportunity for workers to reflect and improve their task performance. Timely and task-specific feedback can help crowd workers learn, persist, and produce better results. We analyze the design space for crowd feedback and introduce *Shepherd*, a prototype system for visualizing crowd work, providing feedback, and promoting workers into shepherding roles. This paper describes our current progress and our plans for system development and evaluation.

Keywords
Crowdsourcing, human computation, feedback system

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Design

re 1: Mechanical Turk is a marketplace small, online tasks. Workers can freely ose which tasks they accept.

tp://www.mturk.com/mturk

tp://turkopticon.differenceengines.com

Introduction

On micro-task platforms like Mechanical Turk[1], requesters pay people to execute short tasks for small amounts of money (Figure 1). Unlike peer-production systems, requesters and workers remain largely anonymous to each other, and little direct interaction occurs between them. Workers can only communicate with other workers through third-party forums[2]. From a labor perspective, treating people as interchangeable replacements for computational processes means that workers often submit assignments with minimal effort [10], and have little opportunity or motivation to improve their understanding of a task domain.

For simple tasks such as data entry, requesters can validate work quality by redundantly hiring workers for the same job [9] or by inserting test problems that have known solutions [10]. However, these strategies are less effective for content-creation tasks — such as writing product reviews, designing advertisements, or categorizing complex data — where requesters desire original and diverse content.

One strategy for accomplishing more complex work is to decompose tasks into iterative or parallel subtasks [3,13]. *Soylent* introduced a find-fix-verify pattern for word processing, where different workers each take on a smaller piece of the larger task [3]. However, within those smaller tasks, an underlying problem persists: workers are not encouraged to learn or improve their performance. *How can crowdsourcing platforms motivate and scaffold novice workers to improve over time, especially on complex, large-scale, creative tasks?* We hypothesize that worker interaction with requesters and with other workers is a key missing component.

In many communities of practice, senior members (often implicitly) help novices learn and stay motivated [12]. Traditional work environments foster employee development through formal performance reviews and feedback, and informally, through peripheral participation [12]. Online communities often provide infrastructure for moderators to review others' content and to encourage the growth of newer members [11]. Peer-production projects like Wikipedia and open-source software have decentralized rather than hierarchical management systems [2]. Individuals choose where to devote resources, and through transparency and reputation systems, the community defines standards and quality control mechanisms [15].

In contrast with traditional firms or peer-production systems, micro-task platforms such as Amazon Mechanical Turk typically offer few formal or informal methods for worker-requester communication. Instructions provide the primary point of contact. The products of crowd workers are invisible to peers. As a result, novice workers cannot observe expert behavior. From a learning perspective, social interaction provides an essential form of feedback [1]. Peer interaction also has motivational benefits [4,8]. LiveOps, a distributed online call center, enabled chat interaction between at-home agents to recreate a "water cooler" setting and to foster cohesion among their workforce [14].

Interactive feedback complements other quality-improvement efforts such as worker qualifications and clearer instructions. We hypothesize that task-specific feedback will help workers on microtask markets improve performance, much as it does in real-world settings, and make workers cognizant that their work is under review. Additionally, feedback may motivate

workers to persevere and accept additional tasks. We investigate these hypotheses through a prototype system, *Shepherd*, that demonstrates how to make feedback an integral part of crowdsourced creative work.

Understanding Opportunities for Crowd Feedback

To effectively design feedback mechanisms that achieve the goals of learning, engagement, and quality improvement, we first analyze the important dimensions of the design space for crowd feedback (Figure 2).

Timeliness: When should feedback be shown?
In micro-task work, workers stay with tasks for a short while, then move on. This implies two timing options: synchronously deliver feedback when workers are still engaged in a set of tasks, or asynchronously deliver feedback after workers have completed the tasks.

Synchronous feedback may have more impact on future task performance since it arrives while workers are still thinking about the task domain. It also increases the probability that workers will continue onto similar tasks. However, synchronous feedback places a burden on the feedback providers; they have little time to review work. This implies a need for tools or scheduling algorithms that enable near real-time feedback. Asynchronous feedback gives feedback providers more time to review and comment on work. However, workers may have forgotten about the task or feel unmotivated to review the feedback and to return to the task.

Design Dimensions for Feedback on Crowdsourced Work

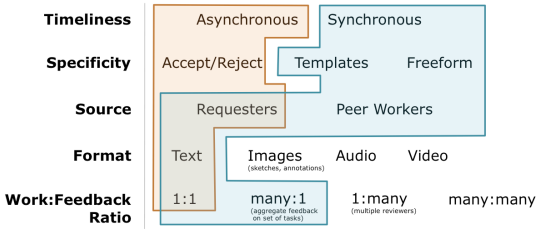


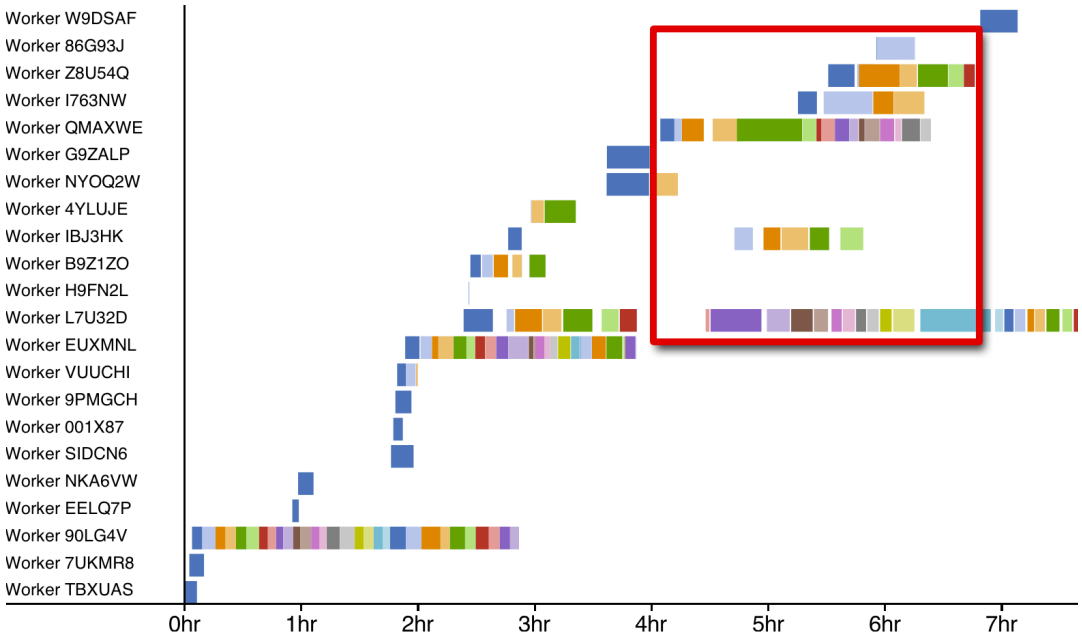
Figure 2: Current systems (in orange) focus on asynchronous, single-bit feedback by requesters. *Shepherd* (in blue) investigates richer, synchronous feedback by requesters and peers.

Currently, platforms like Mechanical Turk only allow asynchronous feedback with no enticement to return. Requesters can provide feedback at payment time, but at that point (typically days later), workers care more about getting paid than improving submitted work. More importantly, unless requesters have more jobs available, workers cannot act on requesters' advice.

Specificity: How detailed should feedback be?
Mechanical Turk currently allows requesters one bit of feedback—accept or reject. While additional freeform communication is possible, it is rarely used unless workers file complaints. Workers may learn more if they receive detailed and personalized feedback on each piece of work. However, this added specificity comes at a price: feedback providers must spend time authoring feedback. When feedback resources are limited, customizable templates can accelerate feedback generation and enable requesters to codify domain knowledge into pre-authored statements. However, templates could be perceived as overly general or repetitive, reducing their desired impact. Workers may need explicit incentive to read and reflect on feedback.

Source: Who should provide feedback?
Crowdsourcing requesters post tasks with specific quality objectives in mind; they are a natural choice for assuming the feedback role. However, experts often underestimate the difficulty novices face in solving tasks [7] or use language or concepts that are beyond the grasp of novices [6]. Moreover, as feedback becomes more specific, requesters may find it more difficult to complete work assessments in real-time.

Alternatively, workers can be paid to provide feedback to other workers. Peer feedback increases scalability as more crowd workers can be recruited to handle the volume of feedback needs. Our preliminary trials indicate that workers perform tasks simultaneously and overlap (see Figure 3). In principle, this overlap opens up the possibility of peer feedback. For example, workers can be promoted into a feedback role after they successfully finish a series of tasks. This introduces the challenge of identifying and promoting knowledgeable and responsible workers.



igure 3: *Shepherd*'s timeline view. Workers overlap in time, which shows potential for peer feedback. This isualization shows work times for 100 product reviews. Rows represent individual workers. The X axis hows time. Each colored bar is one product review. The red rectangle highlights a time segment with ignificant overlap: multiple workers are active simultaneously.

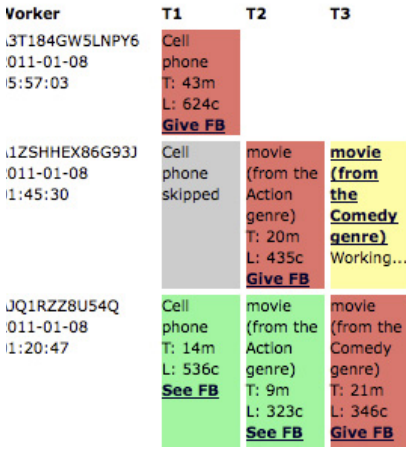
***Shepherd*: System Design**

We are developing *Shepherd*, an infrastructure for managing and providing feedback to crowd workers. Our vision is to make targeted feedback a core component of future micro-task platforms. Requesters will need interfaces to simultaneously author the task and associated feedback form. To administer feedback, requesters will need tools for visualizing work progress. The system will need to elegantly present feedback to workers and confirm that they see and understand the feedback. Also, the system should help requesters decide which workers to promote into advanced roles.

Current Progress

Our prototype recruits and pays workers through Amazon Mechanical Turk; task hosting and data collection occurs on our own Web server. *Shepherd* displays an overview of workers and results in real-time. The *timeline* view (Figure 3) presents a Gantt chart showing when workers accept a task, the length of time workers spend on each task, and how many tasks a worker completes within a batch. In the *matrix* view (Figure 4), columns show tasks and rows show workers. Each box shows the current state of a task (skipped, in progress, finished & needs feedback, or feedback applied).

Requesters can monitor incoming work and click on any task to provide feedback using specially designed forms (Figure 5). To streamline the process, the requester checks high-level feedback categories and the worker receives corresponding critique statements. By default, the system delivers feedback just before a worker begins a new task from the same batch. The choice about timing and delivery method is an empirical question, and depends on factors such as task type and scale.



gure 4: *Shepherd*'s matrix view for a atch of product review tasks. Each box presents the current state of a task. asks can be completed in parallel by ultiple workers (rows). Red boxes dicate tasks are ready for review. ellow boxes are tasks in progress. reen boxes indicate that work is nished and feedback provided. Grey oxes show tasks that workers choose to

Future Development

Micro-task platforms typically provide task authoring templates. *Shepherd* will give requesters tools for specifying feedback forms in tandem with task creation. Feedback templates become especially important when workers review others' work. We will evaluate the overhead costs for creating feedback templates in addition to the task.

A workforce administration interface will let requesters promote/demote workers to shepherding roles, track worker performance over time, and launch tasks for specific workers under controlled criteria. An inference algorithm will recommend promising workers based on prior task performance and domain knowledge ascertained from short interspersed test questions.

Preliminary Evaluation and Future Studies

To study the effects of feedback on crowdsourcing, we will choose tasks that fulfill three key criteria:

- The task domain should have some precedence to ensure relevance to the crowdsourcing community.
- Tasks should have open-ended solutions, so expert feedback has the potential to improve results.
- Results should be objectively measurable to understand if our manipulations affect work quality.

In our preliminary experiments, participants designed Web advertisements [5]. Workers generated ideas for graphical or text-based web ads that were later posted online through Google AdWords to garner click-through rates. Our first study utilized independent raters to judge quality.

(How) does synchronous accept-reject feedback improve task performance and worker satisfaction?

In a between-subjects study, participants generated short advertising phrases for a common client. They could write multiple catchphrases (up to ten) for $0.10 each; they could leave the task at any time. We recruited 58 participants from Amazon Mechanical Turk into two conditions (27 got feedback; 31 got none). In the feedback condition, participants saw their prior catchphrase ideas in a list. Within a minute after submission, the catchphrase would be labeled as either Accepted or Rejected. In the no-feedback condition, participants saw their catchphrases appear in a list, but did not receive Accept or Reject feedback. Participants submitted an average of 4.2 catchphrases for a total of 243 unique ad ideas. Ten independent judges rated each idea on a 20-point scale—accounting for theme, creativity, and professionalism—providing 2430 independent ratings.

An analysis of variances was performed with condition (Feedback and No feedback) and independent judge (raters 1-10) as factors and performance rating as a dependent variable. The Feedback condition (μ=9.4, SD=4.7) outperformed the no feedback condition (μ=8.9, SD=4.6) ($F(1,2429)=8.65$, $p<0.05$). The No feedback condition produced more ideas overall (138 to 105), but the difference was not significant. While overall ratings differed significantly between judges ($F(9,2421)=10.41$, $p<0.05$), all but one agreed that Feedback ads were better.

gure 5: The *Shepherd* feedback form r product reviews collects numerical tings and enables requesters to rapidly semble written feedback from a set of e-authored statements.

Does requester feedback outweigh self-review?
Even simple binary feedback improves results, but do detailed assessments lead to further performance increases? Does the added cost of assessing work outweigh simpler mechanisms such as asking workers to assess their own work? Our next experiment will compare requester-provided and self-report assessments. Participants will write customer reviews for products or services. In this common crowdsourcing task, workers can potentially benefit from expert feedback. We can measure performance by hosting reviews on product sites and measuring community feedback on their helpfulness. This upcoming study will also analyze overhead costs associated with providing feedback; worker self-assessments may lead to cheaper performance gains.

Can workers be effective shepherds?
Longer term, we want to investigate the potential of recruiting workers to provide feedback for other workers on a large-scale content-creation project. We will study differences in how workers and requesters confer feedback and examine the effects of the presentation, source, and tone of feedback.

References

1. Annett, J. Feedback and human behaviour: the effects of knowledge of results, incentives, and reinforcement on learning and performance. Penguin Books, 1969.
2. Benkler, Y. Coase's Penguin, or, Linux and "The Nature of the Firm." The Yale Law Journal 112, 3 (2002), 369-446.
3. Bernstein, M.S., Little, G., Miller, R.C., et al. Soylent: a word processor with a crowd inside. Proc. 23nd ACM Symp. on UIST, ACM (2010), 313–322.
4. Cheshire, C. and Antin, J. The Social Psychological Effects of Feedback on the Production of Internet Information Pools. Journal of Computer-Mediated Communication 13, 3 (2008), 705-727.
5. Dow, S.P. Using Crowds to Study Creativity. Crowd-Conf, (2010).
6. Ericsson, K.A. and Smith, J. Toward a General Theory of Expertise: Prospects and Limits. Cambridge University Press, 1991.
7. Hinds, P. The Curse of Expertise: The Effects of Expertise and Debiasing Methods on Predictions of Novice Performance. Journal of Experimental Applied Psychology 5, (1999), 205-221.
8. Horton, J.J. Employer Expectations, Peer Effects and Productivity: Evidence from a Series of Field Experiments. SSRN eLibrary, (2010).
9. Ipeirotis, P.G., Provost, F., and Wang, J. Quality management on Amazon Mechanical Turk. Proc. ACM SIGKDD HCOMP, ACM (2010), 64–67.
10. Kittur, A., Chi, E.H., and Suh, B. Crowdsourcing user studies with Mechanical Turk. Proc. 26th ACM SIGCHI Human Factors (2008), 453-456.
11. Lampe, C. and Resnick, P. Slash(dot) and burn: distributed moderation in a large online conversation space. ACM SIGCHI Hum. factors (2004), 543–550.
12. Lave, J. and Wenger, E. Situated Learning: Legitimate Peripheral Participation. Cambridge, 1991.
13. Little, G., Chilton, L.B., Goldman, M., and Miller, R.C. TurKit: tools for iterative tasks on mechanical Turk. Proc. ACM SIGKDD Workshop on HCOMP, (2009), 29–30.
14. Musico, C. There's No Place Like Home. destinationCRM.com, 2008.
15. Viégas, F., Wattenberg, M., and Mckeon, M. The Hidden Order of Wikipedia. In Online Communities and Social Computing. 2007, 445-454.