# Mini-Crowdsourcing End-User Assessment of Intelligent Assistants: A Cost-Benefit Study

Amber Shinsel[1], Todd Kulesza[1], Margaret Burnett[1], William Curran[1], Alex Groce[1], Simone Stumpf[2], Weng-Keen Wong[1]

[1]Oregon State University
Corvallis, U.S.A.
{shinsela, kuleszto, burnett, alex, curranw, wong}@eecs.oregonstate.edu

[2]City University London
London, U.K.
Simone.Stumpf.1@city.ac.uk

*Abstract*—**Intelligent assistants sometimes handle tasks too important to be trusted implicitly. End users can establish trust via systematic assessment, but such assessment is costly. This paper investigates whether, when, and how bringing a small crowd of end users to bear on the assessment of an intelligent assistant is useful from a cost/benefit perspective. Our results show that a mini-crowd of testers supplied many more benefits than the obvious decrease in workload, but these benefits did not scale linearly as mini-crowd size increased—there was a point of diminishing returns where the cost-benefit ratio became less attractive.**

***Keywords: end-user programming; testing; crowdsourcing; machine learning***

## I. Introduction

Intelligent assistants customize their work around an end user's needs—they learn how to recognize everything from junk e-mail to photos of friends. These assistants are taking on increasingly critical roles, such as assisting in qualitative research [11]. Work like this may be too important to blindly trust to an assistant, particularly since even well trained assistants are not 100% reliable.

This paper focuses on assistants that serve small groups of people—a smart home security system may serve a family in their home or the tenants of an apartment building, a classifier for a department's electronic bulletin board serves those employees, and a research "coding" assistant helps the group of researchers working on a project.

We refer to these groups as "mini-crowds" rather than "teams" because assessing the assistant is rarely (if ever) an individual user's primary task. Since assessment itself is not the user's goal, we expect that when users *do* test the assistant's reliability, they test only enough to meet their own objectives. Group members may not know each other (such as the tenants of an apartment complex), and they may work asynchronously, only assessing the assistant when necessary. These groups share more traits with the anonymous crowds associated with crowdsourcing than the teams and workgroups studied in computer-supported cooperative work (CSCW) research, yet are much smaller than what we traditionally think of as a "crowd".

To enable *individual* end users to assess intelligent assistants, we recently introduced WYSIWYT/ML [12] to support systematic testing of an assistant's overall accuracy and to help testers understand the kinds of mistakes their assistant may make. Assessing an intelligent assistant is different from in-house testing, beta testing, and user product reviews. In-house testing is done before deployment, while beta testing and product reviews are done on a fixed version of software. Intelligent assistants, however, continually change as they learn new behavior from their users. WYSIWYT/ML allows end users to assess this evolving behavior.

Using WYSIWYT/ML, end users were able to test more than half of an assistant's work on about 200 items in only 10 minutes. This efficiency is encouraging, but most users failed to find all of the assistant's errors. Could testing with a mini-crowd produce more systematic and cost effective results?

The benefits of this idea seem obvious at first, but people's time is not free. Thus, it is important to weigh the benefits of mini-crowdsourcing a testing effort against the costs of involving an increasing number of people in the task.

This paper presents an empirical study considering mini-crowdsourcing from a cost-benefit perspective. We compare the attitudes and testing outcomes of end users working alone against those working with three mini-crowds to answer three research questions:

RQ1: Finding errors: Can mini-crowds help end users find more of their assistant's errors than they would find working alone? What costs are associated with distributing this error finding among a crowd?

RQ2: Behavior changes: How do users' behaviors and attitudes change in the presence of a mini-crowd?

Do they find the crowd reliable?

RQ3: Greatest benefits at lowest costs: What is the interaction between the benefits derived from a mini-crowd and the costs of employing one? Are there situations where the benefits are eclipsed by the costs?

## II. Related Work

Crowdsourced software testing by end users has not been investigated empirically before, but research in nearby applications suggests the idea has merit. For example, a recent study pointed to potential benefits from crowdsourcing software testing by professional developers [16]. Formative research has also identified benefits of crowdsourcing assessment tasks such as document relevance [5] and machine-assisted language translation [1]. An investigation of reCAPTCHA, which applies collaborative assessment to the digitization of print media, found that when as few as five end users agreed with one another, their collective answer was correct 96% of the time [20]. A study of Mechanical Turk's suitability for usability assessment found that crowds of end users have great potential for rapidly collecting user measurements at a low cost [8]. A similar study of Mechanical Turk's viability for assessing visualization design showed that crowdsourcing could contribute new insights for such designs [7].

Like the above scenarios, testing an intelligent assistant involves a series of user assessments (one for each of the assistant's predictions). Intelligent assistants, however, may change their predictions as they learn from a user's behavior, and the number of potential users is likely to be much smaller (assistants may be shared among a family, building, or workgroup) than in traditional crowdsourcing.

Closely related to software testing are the concepts of software debugging and software comprehension: a programmer must test software to identify failures, understand the software's logic (e.g., its source code) in order to find and fix the faults responsible, and then test the software again to verify that the failures have been resolved. There is recent work supporting end-user debugging and comprehension of intelligent assistants that allows users to interactively correct their assistants. Examples include *why...* and *why not…* descriptions of the assistant's logic [10, 14] and visual depictions of the assistant's correct predictions versus its failures [19]. As a basis for creating explanations, researchers have also investigated the types of information users want before assessing the trustworthiness of an intelligent agent [4, 11]. Recent work by Lim and Dey has resulted in a toolkit for applications to generate explanations for popular machine learning systems [15], and a few systems add debugging capabilities to explanations [10, 11]. Supporting testing of intelligent assistants—with or without crowdsourcing—is a necessary component for supporting explanation and debugging approaches like these.

The crowdsourced testing platform used in this paper's experiment extends WYSIWYT/ML, a non-crowdsourced approach to systematic software testing for end users of intelligent assistants [12]. Systematic testing for end users (without crowdsourcing) was pioneered by the WYSIWYT (What You See Is What You Test) approach for spreadsheet users [18]. WYSIWYT/ML applies many WYSIWYT design principles to the problem of assessing intelligent assistants, leveraging statistical properties of the assistant's behavior to reveal likely software failures. WYSIWYT/ML also adapts traditional software testing concepts, such as test case selection/prioritization and coverage metrics [2], to the domain of intelligent assistants. This paper evaluates the benefits and costs of adding mini-crowds to this end-user testing context.

## III. Experiment Design

To examine the effects of mini-crowds on end-user testing, we designed an experiment that let end users test an intelligent assistant with support from three different crowd sizes. Participants worked with an assistant that automatically classified textual messages and were asked to find all of the assistant's mistakes and to estimate the assistant's overall accuracy.

### A. *Participants and Procedures*

We randomly selected 48 participants from a pool of responses to a campus-wide recruitment notice. These participants were all university students (21 male and 27 female) with little or no programming experience, and none were computer science majors.

To investigate the effects of different sizes of mini-crowds, we established four treatments: Treatment 0 had no crowd (the participant worked alone); Treatment 1 had one other user's tests present; Treatment 6 had six other users' tests present; and Treatment 11 involved eleven other users' tests.

We used a within-subject design where each participant worked with every treatment and message set. The message sets (holding 194 to 199 messages each) were chosen from the well-studied "20 Newsgroups" corpus of public newsgroup postings [9]. We pre-trained the assistant such that it was able to predict each message set with 85%-88% accuracy, as defined by the "gold standard" (the topic assigned by the message's original author, e.g. the topic "Cars" for messages posted to the *rec.autos* newsgroup). The pairing and the ordering of the treatments and message sets was balanced via a Graeco-Latin square (a composition of two orthogonal Latin squares where every row and column contains each element exactly once).

We began by verbally introducing participants to the concept of testing an intelligent assistant's predictions. A researcher then led participants through a 20-minute hands-on tutorial to acquaint them with the features of the software. The tutorial covered basic usage instructions only; it did not detail software testing strategies or approaches.

During the experiment, participants viewed messages the assistant had classified into one of four topics (Computers, Religion, Cars, and Motorcycles). Participants had 10 minutes with each treatment to test as many of these topic predictions as they could, deciding which were correct and which were not. (The 10-minute limit allowed us to see how well time-pressured users could assess their assistants.) After each treatment, participants answered the NASA-TLX questionnaire [6] to assess their attitudes. The experiment ended with a final questionnaire asking participants about their behaviors and responses to the crowds' judgments.

## B. *The Mini-Crowd*

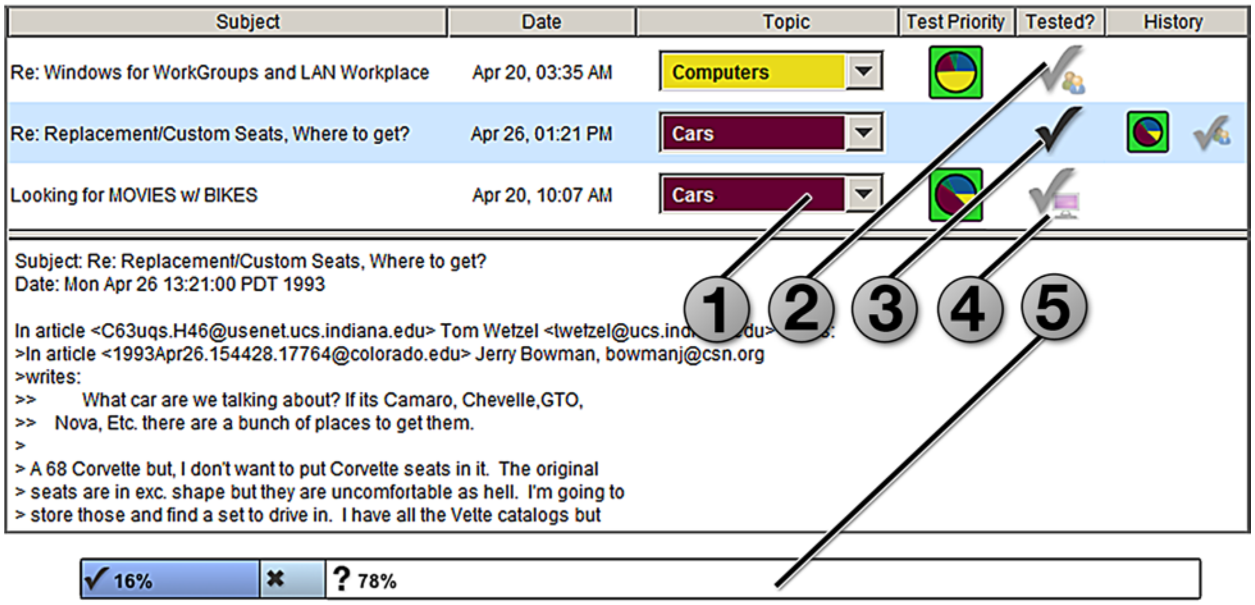
Our study used an *asynchronous* crowd, representing a group of end users working at different times. The crowd's tests were visible to participants for the full duration of each treatment and remained static during the experiment.

To obtain the mini-crowd's judgments, we started with the work of all 12 participants (university students without programming backgrounds, each of whom was compensated for his or her time) from a previous "work-alone" study [12] involving the same testing tool features, message sets, and time limits as the current study. We discarded one outlier because his performance was so much higher than everyone else's (almost 3 times as high) that including his work would have left our new participants with little to do. The 11 remaining participants constituted the mini-crowd for Treatment 11. We then selected the median performer as the "partner" for Treatment 1; and used every second participant (when ordered by performance) for the mini-crowd of 6 (Treatment 6). This method assured that both good and bad testers were equally represented in all treatments.

## C. *Software Environment*

E-mail interfaces are well understood by end users and already involve intelligent assistants (e.g., SPAM filters), so our prototype was designed to mimic an e-mail reader. The environment had five components: (1) the "assistant" itself, a machine learning classifier powered by LibSVM [3] for predicting message topics; (2) a mini-crowd (the size of which varied by treatment) that had previously tested the assistant; (3) an interface enabling participants to test each topic prediction; (4) reasoning devices to notify the user whether WYSIWYT/ML, the participant, or the crowd had tested each message; and (5) devices to explain how much of the assistant's logic had been assessed.

These five components are shown in Figure 1. The top panel included each message's subject, date, and predicted topic (component 1). The mini-crowd's tests appeared as in Figure 1's component 2. A participant could judge the assistant's predictions as *right*, *maybe right*, *maybe wrong*, or *wrong* using the widget in Figure 2, and their judgment would appear in the



| Subject | Date | Topic | Test Priority | Tested? | History |
|---|---|---|---|---|---|
| Re: Windows for WorkGroups and LAN Workplace | Apr 20, 03:35 AM | Computers | | | |
| Re: Replacement/Custom Seats, Where to get? | Apr 26, 01:21 PM | Cars | | | |
| Looking for MOVIES w/ BIKES | Apr 20, 10:07 AM | Cars | | | |

Subject: Re: Replacement/Custom Seats, Where to get?
Date: Mon Apr 26 13:21:00 PDT 1993

In article <C63uqs.H46@usenet.ucs.indiana.edu> Tom Wetzel <twetzel@ucs.indiana.edu> writes:
>In article <1993Apr26.154428.17764@colorado.edu> Jerry Bowman, bowmanj@csn.org
>writes:
>> What car are we talking about? If its Camaro, Chevelle,GTO,
>> Nova, Etc. there are a bunch of places to get them.
>
> A 68 Corvette but, I don't want to put Corvette seats in it. The original
> seats are in exc. shape but they are uncomfortable as hell. I'm going to
> store those and find a set to drive in. I have all the Vette catalogs but

✓ 16% ✖ ? 78%

Figure 1. The software prototype participants worked with. (Component 1) The classifier's predicted topic. (Component 2) The crowd's decision about this prediciton (size of mark shows how many of the crowd members voted this way). (Component 3) The user marked this prediction as correct, using the affordance in Figure 2. (Component 4) WYSIWYT/ML infers user tests to similar messages. (Component 5) A *test coverage bar* informs users how many of the assistant's predictions have been judged (by the user, the crowd, or WYSIWYT/ML), as correct (√) or incorrect (X).

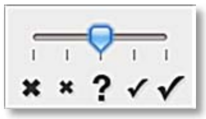


Figure 2. A prediction could be marked as *wrong (X)*, *maybe wrong (x)*, *maybe right (√)*, *right (√)*, or "?" to revert to untested.

*Tested?* column (Figure 1, component 3). Instead of marking *wrong*, a participant could fix the topic using the drop-down topic list (Figure 1, component 1)—this acted as a shortcut for *wrong*, followed by the topic change, followed by *right*. After each participant test, WYSIWYT/ML attempted to infer more tests of similar messages (Figure 1, component 4) [12].

If the participant tested a message that had already been tested by the mini-crowd, the system picked the participant's test to display. Otherwise, the system showed the mini-crowd's tests (if there were any), or WYSIWYT/ML's inferred test otherwise. If members of the crowd disagreed, the displayed check- and X-marks were scaled such that the more common response was larger.

WYSIWYT/ML's responsibilities were to prioritize which predictions participants should test, infer additional tests, measure test *coverage* (how much of the assistant's logic had been tested), and track this coverage over time [12]. We summarize its reasoning here.

To prioritize which predictions participants should test, WYSIWYT/ML used the assistant's confidence in each prediction, and communicated this priority via a green square's brightness (*Test Priority* column in Figure 1). A pie graph shows *how* this priority was computed—the size of each slice represents the assistant's estimated probability that the message belonged in the topic associated with that color.

To show test coverage (how many predictions had been tested), WYSIWYT/ML maintained the progress bar shown in Figure 1, component 5. This bar was updated after each interaction, allowing participants to see how much of their assistant had been tested. The *History* column (Figure 1, right) showed participants the previous two testing priorities and judgments for each prediction, allowing them to spot changes (if any) in response to their tests. To aid in their problem-solving, participants could sort messages by any column (*Subject*, *Date*, …). For consistency with other e-mail systems, the initial sort order was by *Date*.

Participants were instructed to use these tools as desired to find as many of the assistant's errors as they could.

## IV. Results

To explore mini-crowdsourced testing, we investigated how participants (and the mini-crowds helping them) found the assistant's errors, the ways in which participants relied on the crowd, and how these results changed with the size of the crowd.

### A. In Crowds we Trust?

#### 1) Errors Found

Did a mini-crowd find more errors than a participant working alone (RQ1)? Figure 3 (light bars) illustrates what one would suspect: that as mini-crowd size grew, the number of detected errors also increased (repeated measures ANOVA, F(3,136)=46.5 $p$<.001). Table 1 reveals that the mini-crowd was largely responsible for this increase. The crowd played an important role in finding the assistant's errors, leaving only a handful undetected at crowd size 11 (Table 1).

The dark bars in Figure 3, however, reveal a more nuanced story. The error-finding benefit of additional testers swiftly shrank as the mini-crowd's size grew. Many crowd members repeatedly found the *same* errors, possibly as a result of working asynchronously. While larger crowd sizes found more errors overall, they did so increasingly inefficiently.

Participants clearly benefited from the mini-crowd's help in identifying the assistant's errors, but RQ2 asks whether participants found the crowd reliable—did they trust the crowd to *correctly* find errors? Questionnaire responses indicate that participants were keeping a close eye on the crowd's work. When asked whether they thought the crowd's judgments were correct, four participants said "No", while 10 said "Yes". A majority (24 participants) responded with "Yes" followed by a qualifying phrase (e.g., "I think they were for the most part…"). These participants appeared to pay attention to the crowd's error-finding successes, as well as its failures.

#### 2) Where the Errors Aren't

One method for finding errors is to eliminate non-errors (i.e., tests that pass) from the user's search space. This can be accomplished with tests *covering* some strategic fraction of the input space.

As described in Section 3, our measure of coverage included tests the user and crowd explicitly performed, as well as messages similar to (and sharing the same

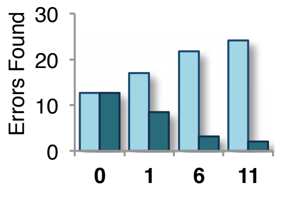


Figure 3. (light) The number of errors identified increased with mini-crowd size. (dark) The benefit/cost ratio decreased as crowd size increased (errors found over cumulative time spent testing).

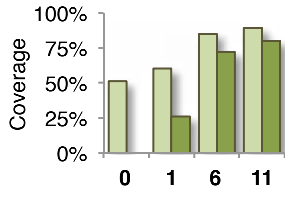


Figure 4. (light) The total coverage went up as crowd size increased. (dark) The crowd heavily contributed to total coverage.

predicted topic as) those explicitly tested. Thus, a single participant test could cover several of the assistant's predictions, allowing users to quickly eliminate these "safe" predictions from their search space.

As Figure 4 shows, participants working alone achieved a mean coverage of 51%, but as crowd size increased, so did total coverage. By crowd size 11, the average coverage was 89%, leaving only 22 untested predictions. Thus, the mini-crowd's tests helped to not only identify errors, but also isolate the areas of the assistant likely to *still* contain errors. By reducing this search space, participants faced a more manageable subset of predictions requiring their attention.

### 3) *Frustration, Engagement, and Reliability*

We have shown that larger mini-crowds helped participants find more errors, but what role did the crowd have on participant attitudes and behaviors?

Participants were involved in approximately the same amount of testing regardless of crowd size, despite the fact that larger mini-crowds identified far more errors (Table 1). The NASA-TLX questionnaire data suggests an explanation—participants reported feeling much less discouraged, stressed, and irritated when testing with a crowd of at least six other end users (Table 2, row Frustration). These participants could not have felt successful because they were individually finding more errors (they were not), but we hypothesize that their attitudes were buoyed by a sense of belonging to a larger group that *was* successfully finding errors.

The above explanation assumes participants trusted the crowd's testing. There is a critical distinction between *responsible* trust versus *blind* trust. The former assumes some amount of verification to confirm that trust is still warranted, while the later represents a form of disengagement, with all decision-making ceded to a third party. To further explore RQ2, we examined how participants formed their judgments about the crowd's reliability.

Table 3 shows that the amount of time participants spent verifying the crowd increased with the mini-crowd's size, suggesting that participants did not blindly trust the mini-crowd. Such verifications seem necessary to fully benefit from the crowd's work, but the link between crowd size and the effort participant's

TABLE I. NUMBER OF ERRORS IDENTIFIED BY EACH ACTOR.

| *Crowd Size* | *Crowd* | *WYSIWYT/ ML* | *Participant* | *Total*[1] | *Remaining* |
|---|---|---|---|---|---|
| 0 | n/a | 2.9 | 11.1 | 12.7 | 15.79 |
| 1 | 9.8 | 3.1 | 10.3 | 17.0 | 10.54 |
| 6 | 19.6 | 2.8 | 10.7 | 21.8 | 5.46 |
| 11 | 22.5 | 2.9 | 12.1 | 24.1 | 4.10 |

[1]Totals do not add up because some errors were jointly identified by multiple actors (e.g., both the crowd and the participant).

expended while forming judgments about the crowd's reliability reveals a cost: it took more effort to establish trust in a larger crowd. Only one participant achieved 100% coverage, so it is unlikely participants were verifying the mini-crowd because they had nothing left to do.

We also observed a link between the size of a crowd and its overall reliability as an oracle—the more people in the crowd, the more their assessments agreed with our gold standard (in the case of disagreements, we used the assessment shared by the majority of testers). This agreement began with an average reliability of 88% for crowd size 1 and rose to 94% for crowd size 11 (ANOVA contrast, F(2,136)=24.8, $p$<.001). Input from more people generally resulted in more reliable error finding.

Thus, the benefits of a larger crowd are significant (RQ3). While participants spent more time verifying crowd tests as mini-crowd size increased, those crowd tests became more reliable. Larger crowds found more errors correctly, reduced the level of stress and frustration experienced by participants, and yet did not leave participants feeling disengaged.

## B. *The Bugs That Got Away*

Even with a mini-crowd's support, some errors remained unseen, or worse, imitated non-errors. We term such incorrect predictions that the crowd nonetheless marked as correct *false negatives*. False negatives reflect bugs still lurking in the assistant's reasoning.

Interestingly, crowd size 6 had the most false negatives, as Figure 5 shows. An ANOVA contrast revealed that the number of false negatives in crowd size 6 is significantly higher than both crowd size 1 and crowd size 11 (Table 2, row False Negatives). We hypothesize that this mini-crowd revealed two nuances

TABLE II. ANOVA CONTRASTS ILLUSTRATING THE DIFFERENCES IN SEVERAL CROWD METRICS BETWEEN TREAMENTS.

| | *Mean value per crowd size (p-value for contrast with shaded cell)* | | | | df | F | p |
|---|---|---|---|---|---|---|---|
| | *0* | *1* | *6* | *11* | | | |
| Errors identified | n/a | 9.8 ($p$<.001) | 19.6 (n/a) | 22.5 ($p$<.001) | 2,136 | 124.2 | <.001 |
| False negatives | n/a | 3.1 ($p$<.001) | 8.2 (n/a) | 6.9 ($p$=.034) | 2,136 | 36.7 | <.001 |
| Coverage | n/a | 26% ($p$<.001) | 72% (n/a) | 80% ($p$<.001) | 2,136 | 1345.0 | <.001 |
| Reliability | n/a | 88% ($p$<.001) | 91% (n/a) | 94% ($p$<.001) | 2,136 | 24.8 | <.001 |
| Frustration (max 21) | 7.4 (n/a) | 6.0 ($p$=.143) | 5.4 ($p$=.042) | 5.3 ($p$=.033) | 3,182 | 1.9 | =.122 |

TABLE III. PERCENT OF PARTICIPANT TESTS THAT VERIFIED EXISTING CROWD JUDGEMENTS.

| *Crowd Size* | *Overlap between participant and crowd tests* | *Disagreements with crowd judgment* |
|---|---|---|
| 1 | 48.7% | 2.3 |
| 6 | 79.2% | 3.9 |
| 11 | 86.7% | 4.4 |

of crowdsourced assessment.

First, crowd size 1 tested fewer predictions than crowd size 6, so the latter had more opportunities to provide false negatives (e.g., if each member of a crowd marks one false negative during their testing, it follows that a larger crowd will have more false negatives overall). If the crowd and problem sizes match up so that each member is working on a unique subset of the task (i.e., no redundancy), then we would expect to see more false negatives introduced as crowd size increases—a dangerous situation that masks bugs remaining in the assistant's logic. Thus, even though crowd size 6 accomplished a high level of coverage (nearly as high as crowd size 11), their judgments were less reliable than the larger crowd's.

The second nuance explains why crowd size 11 did not introduce more false negatives than crowd size 6; as our mini-crowd grew, the problem size remained constant. Thus, an erroneous judgment by one crowd member had ample chance to be overruled by a majority of others in the crowd; their redundant testing helped to isolate false negatives. Our crowd size 6 had little testing overlap, so majority decisions did not effectively safeguard judgment reliability.

The opposite of a false negative is a *false positive*, i.e., the crowd believes a correct prediction is actually an error. False positives waste users' time by forcing them to analyze purportedly incorrect predictions that were correct all along. The number of false positives was reasonably small, averaging between 3 and 4 for each crowd size. This is fortunate—a large number of such errors could reduce the testing system's effectiveness to that of ad hoc testing, wherein the end user has no systematic method for quickly identifying errors.

The number of false positives did not follow the same parabolic trend as false negatives. This may be

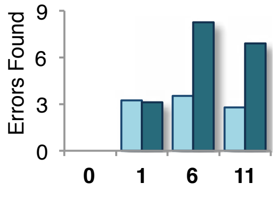


Figure 5. (light) False positives remained constant across treatments. (dark) Larger mini-crowds introduced more false negatives, especially size 6.

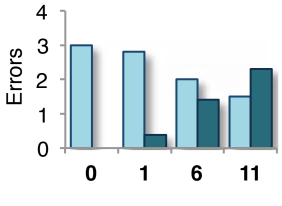


Figure 6. (light) Smaller mini-crowds left more surprise errors hidden in the assistant. (dark) Larger crowds found more of these errors.

the result of positive test bias [17] (the phenomenon where a user tends to choose tests that confirm their own hypotheses, rather than refute them). This theory suggests that if our participants were unsure of a topic prediction, they were more likely to mark the test as a correct prediction than an error.

Besides false negatives and false positives, a third influence may have allowed some bugs to slip through the cracks: *surprise errors*. A surprise error occurs when an assistant is at least 80% confident about an incorrect prediction. These errors are challenging to find with systems like WYSIWYT/ML, which attempt to guide users toward mistakes using (among other methods) the assistant's confidence in each prediction. Crowd size had a positive effect on the number of surprise errors found (Figure 6). As with false negatives and false positives, our largest mini-crowd yielded the greatest benefit to participants.

### *C. How Many Eyes?*

#### *1) The Point of Diminishing Returns*

We have shown that when testing an intelligent assistant, a crowd led to increased test coverage and error finding without additional participant effort. We also revealed situations where a larger crowd was associated with increased costs, such as the high number of false negatives from crowd size 6, or the amount of effort required to establish trust in a larger crowd. At some point, the benefit to end users may not increase proportionally with crowd size, so it is useful to identify factors leading to a point of diminishing returns (RQ3).

Recall Figures 3 and 4: both show that while the number of errors found and test coverage grew with crowd size, neither increased linearly. There are large improvements from crowd size 0 to crowd size 1, and again to crowd size 6. When the crowd size was increased to 11, however, the increases in both errors found and test coverage were markedly smaller.

Why these diminishing returns? Unlike many domains where crowdsourcing has been successfully employed (such as language translation or word recognition, e.g. [20]), both of our performance metrics (errors found and logic covered) have respective upper bounds. Thus, adding more crowd members implies there *must* be a point of diminishing returns—eventually, all of the errors in the assistant's current predictions will be uncovered. In a problem domain where the measure of a crowd's work is bounded, adding more workers is unlikely to increase this measure linearly. For example, we ran offline experiments to assess our initial crowd coverage on crowd sizes from one to 11. Our findings showed large jumps in coverage at sizes two and six, with much smaller increases for other crowd sizes.

Redundancy is another source of diminishing returns. In our study, time was limited to 10 minutes, preventing any single participant from individually testing each of the assistant's predictions. With the addition of a mini-crowd, however, participant tests began to overlap with crowd tests. This overlap became more severe as crowd size increased. In fact, the crowd itself frequently duplicated its own testing efforts, as is clear from the bars representing errors found per time spent testing (Figure 3).

As Section 4.B revealed, a major benefit of having more eyes searching for errors was not simply finding possible errors, but *correctly* identifying these mistakes. Clearly, the value of redundancy will be domain specific—when a family assesses their smart home security system, there is little value to having multiple confirmations of a security threat, whereas co-workers evaluating a shared SPAM filter may have very different ideas of which messages are undesirable. We will discuss methods that may alleviate this duplication of work in Section 5.

### 2) *Leveling the Playing Field*

Prior research in end-user testing has uncovered various stratagems employed by end users [12], such as focusing on high priority tests to find errors or leveraging WYSIWYT/ML's inferred tests to cover most of the assistant's logic. The use of these stratagems led some participants to notably outperform their peers, but encouraging the adoption of strategies remains an open research area. Perhaps a mini-crowd of end users working together could level the playing field; each user brings his or her own talents to the task, and everyone shares the benefits.

Participants in our study had the option of sorting the assistant's predictions on WYSIWYT/ML's Test Priority, which we know from earlier work [12] is a strategy resulting in successfully finding most of the assistant's errors. The vast majority of participants who used this sort method at all used it for most of the experiment, with a median of 4 minutes and 55 seconds. We split our results into two groups based on whether the participant used this Priority stratagem for more or less than the median time. Figure 7 shows the number of errors found by participants who used the Priority stratagem (dark bars) and those who did not (light bars). The differences in crowd size 0 and crowd size 1 are particularly striking.

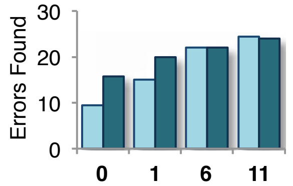


Figure 7. (light) Participants who avoided the Priority stratagem needed the mini-crowd's help to find as many errors as other participants (dark).

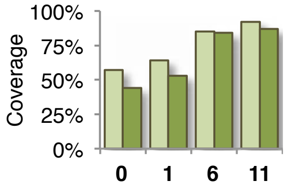


Figure 8. Participants who used the Priority stratagem (dark) needed the mini-crowd's help to achieve similar coverage as other participants (light).

The advantage, especially at the smallest crowd sizes, of using the Priority stratagem has two implications. First, using the WYSIWYT/ML-supported stratagem of Priority was approximately as good as having a helpful mini-crowd the next size larger, providing a low-cost way of achieving the same benefit as with a mini-crowd. Second, after a certain small size (here, 6), a mini-crowd was able to compensate for lack of good strategies. Thus, in situations where convincing end users to employ certain strategies is difficult or unrealistic, mini-crowdsourced testing may be able to achieve an equivalent level of accuracy.

Our test coverage measure (Figure 8) reflects this "leveled playing field" in reverse—participants employing the Priority stratagem averaged *less* coverage than others, until increasing crowd sizes mitigated the effect. Prior work [12] also found that participants employing the Priority stratagem achieved lower coverage than other participants, likely because high priority tests were, by a number of different measures, extremely unlike anything in the assistant's training set. Because similarity to the training set was used to determine which similar predictions would be "covered" by each participant test, it logically follows that focusing on high priority tests would result in lower coverage. Figure 8, however, reveals very little difference in coverage obtained by both groups of participants at larger crowd sizes. We hypothesize that a crowd is likely to include testers using different strategies, imparting the benefits of each strategy to everyone in the crowd and minimizing each strategy's drawbacks. This is particularly evident when the strategies are complementary (as in this study).

## V. Discussion

A common concern regarding group tasks is the phenomenon of "social loafing" [13]. In essence, this theory predicts people are likely to be more productive on their own than in a group, with the size of the group predicting the drop in productivity. Critically, we found no evidence of social loafing among our participants. Our results in Section 4.A.3 reveal that participants remained engaged throughout each task, suggesting that, while other costs (such as decreased efficiency) should be weighed against the benefits of mini-crowdsourced testing, decreasing productivity is not one of them.

The duplication of effort by an asynchronous crowd is another common concern. As discussed in Section 4.B, this redundancy increased oracle reliability, but at some point redundancy seems unlikely to provide a benefit worth its cost. For example, [20] found that once five people agreed about an assessment, they were correct 96% of the time. To help address this issue, it may be possible to control redundancy. For example, WYSIWYT/ML could prioritize tests based on crowd *disagreement*, so that each crowd member will have more opportunities to assess areas of the assistant that have conflicting tests. Conversely, tests with a high level of agreement could have a very low testing priority, thereby redirecting user effort toward the areas it will be most beneficial.

Finally, our focus has been on *anonymous* mini-crowds, but there are situations where members of a mini-crowd may be quite familiar with one another. A promising area for future work involves building upon computer-supported collaborative work (CSCW) research to support small groups of collaborating end users assessing a shared assistant, such as with family homes or small workgroups.

## VI. Conclusion

This paper provides the first empirical evaluation of mini-crowdsourcing the assessment of intelligent assistants. As these assistants take on more critical tasks, assessing when to rely on them will become increasingly important. Our results show that using an asynchronous mini-crowd to assess these assistants confers benefits to end users, but not without costs. This paper has empirically investigated the trade-offs to better understand the "price" of these benefits.

Larger mini-crowds, as expected, found more of an assistant's errors, tested more of its logic, and introduced enough redundancy to reduce crowd mistakes, as compared with smaller mini-crowds. However, results we did *not* expect were:

- Bigger was not always better: the mini-crowd of 6 was *worse* about introducing false negatives than the mini-crowd of 11.
- Diminishing returns: even in metrics where larger mini-crowds outperformed smaller crowds, the benefit of increasing the crowd size quickly dropped, while the cost scaled linearly.
- No loafing: contrary to the phenomena of social loafing, participants working with large mini-crowds did not overly rely upon the crowd.
- Tool-supported strategies versus mini-crowds: participants using the WYSIWYT/ML-supported "priority" strategy found as many errors as participants working with larger mini-crowds.

Overall, our results are encouragingly positive about a future in which shared testing is paired with shared debugging, to support small ecosystems of end users to quickly and effectively assess intelligent assistants that support important aspects of their work and lives.

## Acknowledgments

We thank our participants, Shubhomoy Das, Travis Moore, Shalini Shamasunder, and Katie Shaw. This work was supported in part by NSF 0803487.

## References

[1] Ambati, V., Vogel, S., Carbonell, J. Active learning and crowd-sourcing for machine translation. *Proc. LREC* (2010), 2169-2174.

[2] Beizer, B. *Software Testing Techniques.* International Thomson Computer Press (1990).

[3] Chang, C. and Lin, C. LIBSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm (2001).

[4] Glass, A., McGuinness, D. Wolverton, M. Toward establishing trust in adaptive agents. *Proc. IUI*, ACM (2008), 227-236.

[5] Grady, C. and Lease, M. Crowdsourcing document relevance assessment with Mechanical Turk. *Proc. NAACL HLT Wkshp. Creating Speech and Language Data with Amazon's Mechanical Turk* (2010), 172-179.

[6] Hart, S. and Staveland, L. Development of a NASA-TLX (Task load index): Results of empirical and theoretical research, Hancock, P. and Meshkati, N. (Eds.), *Human Mental Workload* (1988), 139-183.

[7] Heer, J. and Bostock, M. Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design, *Proc. CHI*, ACM (2010), 203-212.

[8] Kittur, A. Chi, E., and Su, B. Crowdsourcing user studies with Mechanical Turk, *Proc. CHI*, ACM (2008), 453-456.

[9] Kniesel, G. and Rho, T. Newsgroup data set. http://www.ai.mit.edu/jrennie/20newsgroups (2005).

[10] Kulesza, T., Wong, W., Stumpf, S., Perona, S., White, R., Burnett, M., Oberst, I., and Ko, A. Fixing the program my computer learned: Barriers for end users, challenges for the machine. *Proc. IUI*, ACM (2009), 187-196.

[11] Kulesza, T., Stumpf, S., Burnett, M., Wong, W., Riche, Y., Moore, T., Oberst, I., Shinsel, A., McIntosh, K. Explanatory debugging: Supporting end-user debugging of machine-learned programs. *Proc. VL/HCC*, IEEE (2010), 41-48.

[12] Kulesza, T., Burnett, M., Stumpf, S., Wong, W., Das, S., Groce, A., Shinsel, A., Bice, F., and McIntosh, K. Where are my intelligent assistant's mistakes? A systematic testing approach, *Proc. IS-EUD (LNCS 6654)*, 171-186.

[13] Latané, B., Williams, K., and Harkins, S. Many hands make light the work: The causes and consequences of social loafing. *J. Personality and Social Psychology*, 37, 6 (1979), 822-832.

[14] Lim, B., Dey, A. and Avrahami, D. Why and why not explanations improve the intelligibility of context-aware intelligent systems. *Proc. CHI*, ACM (2009), 2119-2128.

[15] Lim, B. and Dey, A. Toolkit to support intelligibility in context-aware applications. *Proc. Int. Conf. Ubiquitous Computing*, ACM (2010), 13-22.

[16] Riungu, L., Taipale, O., and Smolander, K. Research issues for software testing in the cloud, *Int. Conf. Cloud Computing Technology and Science*, IEEE (2010), 557-564.

[17] Ruthruff, J., Prabhakararao, S., Reichwein, J., Cook, C., Creswick, E., and Burnett, M. Interactive, visual fault localization support for end-user programmers. *J. Visual Languages and Computing*, Volume 16 (2005), 3-40.

[18] Rothermel, G., Burnett, M., Li, L., Dupuis, C., and Sheretov, A. A methodology for testing spreadsheets. *ACM Trans. Software Engineering and Methodology 10*, 1 (2001), 110-147.

[19] Talbot, J., Lee, B., Kapoor, A. and Tan, D.S. EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. *Proc. CHI*, ACM (2009), 1283-1292.

[20] Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321 (2008), 1465-1468.