CrowdGrader: A Tool For Crowdsourcing the Evaluation of Homework Assignments*

Luca de Alfaro
Computer Science Dept.
University of California
Santa Cruz, CA 95064, USA
luca@ucsc.edu

Michael Shavlovsky
Computer Science Dept.
University of California
Santa Cruz, CA 95064, USA
mshavlov@ucsc.edu

ABSTRACT

CrowdGrader is a system that lets students submit and collaboratively review and grade homework. We describe the techniques and ideas used in CrowdGrader, and report on the experience of using CrowdGrader in disciplines ranging from Computer Science to Economics, Writing, and Technology. In CrowdGrader, students receive an overall *crowd-grade* that reflects both the quality of their homework, and the quality of their work as reviewers. This creates an incentive for students to provide accurate grades and helpful reviews of other students' work. Instructors can use the crowd-grades as final grades, or fine-tune the grades according to their wishes. Our results on seven classes show that students actively participate in the grading and write reviews that are generally helpful to the submissions' authors. The results also show that grades computed by CrowdGrader are sufficiently precise to be used as the homework component of class grades. Students report that the main benefits in using CrowdGrader are the quality of the reviews they receive, and the ability to learn from reviewing their peers' work. Instructors can leverage peer learning in their classes, and easily handle homework evaluation in large classes.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science**]: Education

Keywords

Crowdsourcing, Grading, Peer evaluation

1. INTRODUCTION

Grading complex assignments, such as essays or computer-science submissions that require compilation or build steps, can be a cumbersome and time-consuming process. Especially for large classes, the grading burden on the instructor and teaching assistants may limit the amount of in-depth feedback that the students receive for their submissions. At the same time, students are often receptive in learning from their peers, and they can benefit from being able to compare and discuss their solutions with other students (see, e.g., [2, 4]).

To explore the use of peer feedback in grading homework, we developed CrowdGrader, a web-based tool for the collaborative grading and evaluation of homework assignments. Students submit their solutions to homework problems to CrowdGrader, and they are then asked to review and grade a small number (usually, 4 to 6) of submissions by other students. The overall grade they receive in a homework assignment depends both on the (aggregate) grade they receive for their submission, and on their effort and precision in reviewing their peer's submissions. Thus, students have an incentive to provide accurate evaluations. CrowdGrader is publicly available at `http://www.crowdgrader.org`. So far, it has been used for assignments ranging in topic from Computer Science, to Economics, Biology, Literature, Technology, and Writing; the class size ranged from a dozen students to many hundreds.

In building CrowdGrader, we hoped students would benefit from being able to examine the solutions submitted by other students: accomplished students would be able to look at alternative ways of solving the same problem, and students who encountered difficulties would be able to study several working solutions to the problem while grading. We also hoped that students would benefit from their peer's feedback, which can be more varied and more detailed than what can be provided by a single instructor, or a few teaching assistants, in charge of a large class.

We describe the ideas and techniques used in CrowdGrader, and we report on the experience and quantitative results in using CrowdGrader in seven classes: four Computer Science classes focused on Android and Web development, algorithms and C++, and Java; one Economics class, and two Engineering classes in which homework consisted in essay-writing.

The lifecycle of an assignment in CrowdGrader consists of three phases: a submission phase, a review phase, and a grading phase.

The submission phase is standard: students can submit their solutions either individually or, if the homework assignment allows it, as groups of collaborating students.

In the review phase, each student must review a given number of submissions. In our experiments, asking that each submission was reviewed by 5 or more students yielded sufficient accuracy, while resulting in acceptable workload for the students. The problem of assigning reviews to students turned out to be more complex than expected. In every assignment, there are students who do not do their reviews, and naive approaches to assigning reviews led to some submissions receiving too few reviews. We eventually resorted to an on-line algorithm, in which students are assigned a review task only upon completing the previous one; the algorithm

*CrowdGrader is a trademark of CrowdGrader LLC. This work was supported in part by the Google Research Award "Crowdsourced Ranking". The authors are listed in alphabetical order.

SIGCSE'14, March 5–8, 2014, Atlanta, GA, USA.

ACM 978-1-4503-2605-6/14/03 ...$15.00.
http://dx.doi.org/10.1145/2538862.2538900.

dynamically estimates the probability that each will be completed, on the basis of previous history, and assigns review tasks to try to achieve uniform coverage.

When students are assigned a submission to evaluate, they must review it, and evaluate its quality. We experimented with two approaches to quantitative evaluation: one based on ranking, the other based on grades. We initially asked students to rank the submissions they reviewed in quality order. We thought that, while students might not always be able to assign absolute grades with precision, they are likely to have a correct opinion of the relative merits of the submissions. This approach did not work well in practice: students instinctively disliked the process, often omitting the ranking step, and they expressed doubts about the accuracy of the final result. Consequently, we switched to grades, asking students to assign grades to the submissions they review. We experimented with several algorithms for aggregating the grades received by each submission into a single *consensus grade* for each submission; some initial results are reported in [6]. For the results given in this paper, we used the algorithm used in Olympic competitions: the lowest and highest grades received are discarded, to help eliminate outliers, and the remaining grades are averaged.

To provide an incentive for students to be accurate in their grades and helpful in their reviews, CrowdGrader assigns to each student an overall *crowd-grade* that combines three grades:

- the *consensus grade* computed for the student's submission;
- an *accuracy grade* that measures the precision of the student in grading submissions;
- a *helpfulness grade* that measures how helpful were the reviews written by the student.

Making the review accuracy and helpfulness part of the overall grade received by the student generates an incentive to provide good quality reviews and evaluations that worked well in practice: the results show both a high participation in the review process, and a high average helpfulness of the resulting reviews. Instructors can either directly use the crowd-grades as grades for the assignment, or they can fine-tune them before assigning them: the instructor is thus always in control of the final assigned grades.

The students perceived the richness of feedback, and the ability to inspect several working solutions of the homework assignments, as the main benefits of CrowdGrader compared to evaluations by a teaching assistant. Instructors can easily grade assignments in large classes, and by having access to all reviews, they can quickly identify how well the students are learning the subject matter.

In the remainder of the paper, we describe in more detail the various phases of the review and grading process, and we describe the experimental results we obtained in using CrowdGrader in several classes.

2. PREVIOUS WORK

The work most closely related in goals to ours is the proposal to crowdsource the review of proposals for use of telescope time by [12], as well as the recent NSF pilot project for reviewing funding proposals [14]. As in those approaches, we also distribute the task of reviewing the submissions to the same set of people who submitted the items to be reviewed. Both for proposals submitted to a specific panel, and for solutions submitted to the same homework assignment, the submissions are on sufficiently related topics that the problem of matching submission topic with reviewer expertise can be disregarded. For proposals, of course, care must be taken to avoid conflicts of interest; our situation for homeworks is relatively simpler. Where the problems differ is that proposal reviewing is essentially a top-k problem: the best k proposals must be selected for funding. Homework grading, on the other hand, is an evaluation problem: each item needs to be graded on a scale. In top-k problems, the most important consideration is precision at the top; mis-ranking items that are far from the top k carries no real consequence. In our evaluation problem, each evaluation carries approximately the same importance, and we do not need to precisely rank students whose submissions have approximately the same quality. While there are techniques that can be applied to both problems, this difference in goals justifies the reliance of [12, 14] on comparisons, and ours on grades.

The works of [12] and [14] discuss incentive mechanisms for reviewers, consisting in awarding a better placement in the final ranking to proposals whose authors did a better job of reviewing. We follow the same approach, but we have the additional constraint that students must find the reviewing work appropriately rewarded with respect to the time it takes. For this reason, we let instructors choose the relative weights of the consensus grade of the submission, versus the accuracy and helpfulness grades of the reviews, so that the grades can properly account for the fraction of time spent in the various activities.

The effect of review incentive on the quality of the ranking is examined in depth in [13]. The main problem, also raised in [12], is that the incentive mechanism makes the grading a "Keynesian beauty contest", where reviewers are rewarded for thinking like other reviewers; in turn, this may encourage a "race to mediocrity", in which non-controversial, blander proposals may fare better than more audacious and original ones. We agree with the authors of [13] that this may be a true problem for proposal review. However, we believe that in the context of homework assignments, the problem may be minor. The helpfulness and accuracy grades do not overly punish students who mis-evaluate a single submission; this gives more leeway to students presented with a homework submission that does not follow the beaten path. We also believe that the less competitive evaluation setting, as compared to a top-k setting, may lessen the problem.

The topic of crowdsourced grading has been discussed also in an influential blog post [5]. The blog post advances the idea that one of the main benefits of crowdsourced grading consists in teaching the skill to analyze and evaluate the work of others, a view we share.

3. THE REVIEW PHASE

The review phase is of primary importance for the accuracy of the generated ranking, and we experimented with several designs.

3.1 Review assignment

CrowdGrader implements an anonymous review process. Since students cannot choose which submissions they review, and they do not know the identity of the submissions' authors, they have limited ability to collude and cause their friends to receive higher grades.

In conferences, it is customary to assign papers to program-committee members in a single batch; each member then has a period of time to read the papers and enter all reviews. CrowdGrader instead assigns submissions for review one at a time: students are assigned a new review task only upon completion of the previous ones. Unlike conference papers, solutions submitted to assignments can be quite similar one to the other: by having students work on one review at a time, we hoped to reduce the likelyhood of students mixing up the submissions and their reviews. Indeed, we received no valid reports of mis-directed reviews. Furthermore, by delaying the assignment of new reviews until students have finished previous ones, CrowdGrader can better ensure that all submissions

receive roughly the same number of reviews, even if some students fail to do any reviewing work. For each submission, CrowdGrader computes the number of *likely reviews,* consisting of the completed reviews, along with the review tasks that had been assigned only a short time before. When assigning reviews, CrowdGrader chooses from submissions having least number of likely reviews.

3.2 Ranking vs. grades

For the first two homework assignment conducted using CrowdGrader, we decided to ask students to rank homework submissions, rather than grade them. Ranking is a simpler problem than grading, since it involves only a relative, rather than absolute, judgement. Consequently, we thought that asking students to rank the submissions they reviewed would lead to more precise results than grading; an extensive body of literature attests to the fact that precise global rankings can be obtained by merging partial rankings [7, 1, 3, 11, 9, 10, 8].

Unfortunately, many students skipped the ranking step, leaving the submission they just reviewed in the default position where it was placed: at the bottom of the rank. To confirm this, we measured the fraction of times that students would rank the just-reviewed submission higher than a previously-reviewed submission. This fraction should have been close to 50%, since submissions are assigned in an order that does not depend on their quality. Instead, in the first assignment this fraction was only 36%. Even after strongly reminding students to provide a ranking, the fraction rose only to 41% in the second assignment that relied on ranking.

Talking to students, we understood that they were skipping the ranking step because of a combination of forgetfulness, and unwillingness. Several students mentioned that they felt uncomfortable with providing a ranking of their peers. They complained about having to rank arbitrarily submissions they considered roughly equivalent, and they considered ranking a blunt instrument, unable to differentiate between the cases of submissions of similar, and widely different, quality. After the second assignment, we decided to base CrowdGrader on grades: this led to markely increased student satifaction.

3.3 Declining evaluations

We discovered early on that it was important to allow students to decline some review tasks, without incurring negative consequences. In our programming assignments, there were many cases in which students were unable to review submissions due to factors beyond their control. In the CS/Android class, their installation of Eclipse and Android SDK occasionally misbehaved in a way that left students unable to load and review the code submitted by other students. In the CS/C++ class, glitches or differences in the build environment occasionally prevented students from compiling and executing the submissions under review. Initially, when students were required to enter a grade to receive credit for their review effort, they entered very low grades for the submissions they could not evaluate. This caused upset among the recipients of the low grades, and it was the largest source of discrepancy among the grades assigned to the same submission. The solution was to let students flag a review task as "declined", omitting the grade, and providing as review an explanation of why they were declining it.

4. GRADE ASSIGNMENT

Once the review phase is over, CrowdGrader computes for each student a *crowd-grade* that is the result of combining three grades: the *consensus grade* received by the submission, and the *accuracy* and *helpfulness grades* reflecting the quality of the student's review work.

4.1 Consensus grade

We experimented with several techniques for aggregating the grades each submission received into a single *consensus grade.* Currently, CrowdGrader computes the consensus grades by relying on a reputation system, which gives more weight to the grades assigned by the students who have a high measured grading accuracy; the details can be found in [6]. For the sake of simplicity, the results presented in this paper are obtained using the well-known technique used in Olympic competitions: for each submission, we discard the highest and lowest grades, and we average the remaining grades. We nickname this technique *maverage,* as it is a mix of median and average methods; compared to average, maverage is more resistant to outliers. In general, if n grades are available, maverage discards the $\lfloor n/4 \rfloor$ lowest and the $\lfloor n/4 \rfloor$ highest grades before averaging.

4.2 Accuracy grade

To quantify the accuracy of each student in assigning grades, we compare the grading accuracy of the student to the grading accuracy of a fully random grader. Precisely, consider a set S of submissions, and U of users (students). Let $R \subseteq S \times U$ be the set of reviews performed, and let $R \circ j = \{i \mid (i,j) \in R\}$ be the set of submissions reviewed by student j, and $i \circ R = \{j \mid (i,j) \in R\}$ be the set of reviewers of submission i.

For $(i,j) \in R$, denote by g_{ij} the grade assigned by j to i, and let $\hat{g}_i$ be the consensus grade of submission i, computed using the maverage method described above. The average square error $\tilde{v}_j$ of a student $j \in U$ with respect to the consensus grades is:

$$v_j = \mathrm{E}\{(g_{ij} - \hat{g}_i)^2\}_{i \in R \circ j} \ .$$

A hypothetical "fully random" grader, who assigns to each submission a grade picked at random from the complete set of assigned grades, would have average square error given by:

$$\tilde{v} = \mathrm{E}\{(g_{ij} - \hat{g}_l)^2\}_{(i,j) \in R, l \in S} \ .$$

We assign to each student j an *accuracy grade* that measures how much better the student is than such a fully random grader using:

$$a_j = 1 - \sqrt{\frac{\min(v_j, \tilde{v})}{\tilde{v}}} \ .$$

4.3 Helpfulness grade

To provide an incentive for students to write helpful reviews, CrowdGrader allows students to rate and leave feedback on each review they receive. The feedback provided on reviews goes some of the way towards providing an incentive to write helpful reviews: all students naturally like to receive praise for their work, rather than having sloppiness or imprecisions pointed out to them. Furthermore, students can rate the reviews they receive, assigning them integer ratings ranging from -2 (incorrect, completely unhelpful) to $+2$ (very helpful), with 0 being the neutral rating. These review ratings are used to compute the *helpfulness grade* $h_j \in [0,1]$ of each reviewer $j \in U$, as follows. Let L_j be the list of review feedbacks received by j (this list may be shorter than the number of reviews performed by j, as some reviews might not have received any feedback). We drop from L_j one of the lowest feedbacks f with $f = \min L$, obtaining L'. For a feedback $f \in \{-2, \ldots, +2\}$, we let the weight of feedback f be $w(f) = 2$ if $f < 0$, and $w(f) = 1$ if $f \geq 0$, so that negative feedback weighs twice as much as positive feedback. We then compute the helpfulness grade of student $j \in U$ via:

$$h_j = \max\left[0, \min\left[c\left(1 + \frac{\sum_{f \in L'} f\, w(f)}{2 \sum_{f \in L'} w(f)}\right)\right]\right] ,$$

Class	N students	N assign.	N req revs	α
CS/Android	68	5	6	25%
CS/Web	78	2	5	25%
CS/C++	102	5	5	25%
CS/Java	22	1	4	25%
Eng/Essay1	55	2	5	2%
Eng/Essay2	232	1	6	15%
Econ	61	6	5	25%

Table 1: Number of students, number of assignments for the class, number of required reviews, and relative weight α of reviews (expressed as percentage), for the various classes considered.

where $0.5 \leq c \leq 1$ is the default helpfulness for reviewers who receive no feedback for their reviews. This equation has been hand-tuned based on experience; we currently use $c = 0.7$. We discard the lowest rating from L in order to prevent tit-for-tat behaviors, in which students who receive a low grade react by labeling the associated review as unhelpful. Students receive a low helpfulness grade only if multiple of their reviews are labeled as unhelpful, with insufficient helpful reviews to mitigate such negative ratings.

4.4 Crowd-grade and final grade

Once the consensus, accuracy, and helpfulness grades have been computed, they are merged into the *crowd-grade* γ_j of student $j \in U$ by giving a weight $\alpha \in [0, 1]$ to the review work of the student. Indicating by M the maximum grade for the assignment, by n_j the number of reviews performed by j, and by N the number of reviews that every student was supposed to complete, we let:

$$\gamma_j = (1 - \alpha)\hat{g}_j + \alpha M \frac{(a_j + h_j)\ \min\{n_j, N\}}{2N}\ .$$

In CrowdGrader, instructors can choose the percentage α for which the review activity of a student enters in the crowd-grade. Many instructors leave α unchanged from its default value of 25%; as we will see in the next section, this value generates a sufficient incentive to perform the reviews, leading to most of them being completed. Once the crowd-grades are computed, the instructor can either assign them directly as final grades, or the instructor can fine-tune them, changing individual grades to correct mistakes occurred during the review phase, as well as re-scaling them to modify the overall grade distribution to better reflect the level of accomplishment of students in the class. This leaves the instructor always in full control of the grades that are assigned to the students.

5. RESULTS

We provide results on the student participation in the review phase, and on the accuracy of the grades computed by CrowdGrader. Our results are drawn from seven classes that have been taught using CrowdGrader as the primary means of grading assignments: four are Computer Science classes on Android, web, C++, and Java; one is an Economics class, and two are Engineering classes which involved essay writing. The assignments in Computer Science classes consisted in programming assignments in which students had to turn in bundles of files that consituted the applications. The assignments in the Economics class involved answering questions in English. The classes in Engineering involved writing summaries and essays on specific topics. For each of these classes, we include in our statistics all the assignments that used CrowdGrader. Table 1 summarizes some basic data about the classes.

Class	N	Perc rev	Min rev	Avg rev	Avg len
CS/Android	5	106%	50%	90%	203
CS/Web	2	95%	70%	96%	463
CS/C++	5	95%	56%	90%	406
CS/Java	1	100%	75%	105%	79
Eng/Essay1	2	78%	40%	77%	130
Eng/Essay2	1	92%	50%	89%	329
Econ	6	98%	82%	101%	163

Table 2: Participation in the CrowdGrader review phase. *N* is the number of assignments for the class. *Perc rev* is the percentage of students who did at least one review, expressed as percentage of students who submitted a solution; this percentage can be greater than 100% since some students may have only reviewed. *Min rev* and *Avg rev* are respectively the minimum and average number of reviews received by submissions, expressed as percentages of the number of reviews required in the assignments. *Avg rev len* is the average length of an individual review, in number of characters.

Assignment	$\|S\|$	RevsDue	MinRevs	AvgRevs
CS/Android hw 1	60	6	2	5.4
hw 2	61	6	2	5.3
hw 3	68	6	0	4.8
hw 4	62	6	6	6.1
hw 5	57	6	5	5.3
CS/C++ hw 1	102	5	0	4.6
hw 2	97	5	3	4.6
hw 3	91	5	4	5.1
hw 4	97	5	3	4.6
hw 5	90	5	4	5.1

Table 3: Number of reviews assigned and performed for the homework assignments that are part of the dataset. $|S|$ is the number of submissions, RevsDue is the number of reviews that each student ought to have done, MinRevs is the minimum number of reviews received by a submission, and AvgRevs is the average number of reviews per submission.

5.1 Quality of review phase

The participation in the review process in these classes is summarized in Table 2. Participation in the review phase was high, and most submissions received a number of reviews that was close to the desired value. The average review length was also reasonably high.

Table 3 provides more detailed data on the CS/Android and CS/C++ classes, giving the minumum and average number of reviews received by submissions in each of their assignments. These were the first two classes conducted using CrowdGrader. In the first three assignments of the CS/Android class and in the first assignment of the CS/C++ class, CrowdGrader used a simple approach that assigned submissions to review uniformly at random among the submissions in need of review. Since not all students did the required review work, this simple review assignment method led to some submissions receiving an inadequate number of reviews. Once the predictive algorithm described in Section 3.1 was adopted in later assignments, CrowdGrader was able to guarantee a sufficient number of reviews for all submissions.

Figure 1 gives the histogram of the review feedback provided by the students in the class Eng/Essay2; the data indicates that students generally found the reviews to be helpful. Eng/Essay2 was

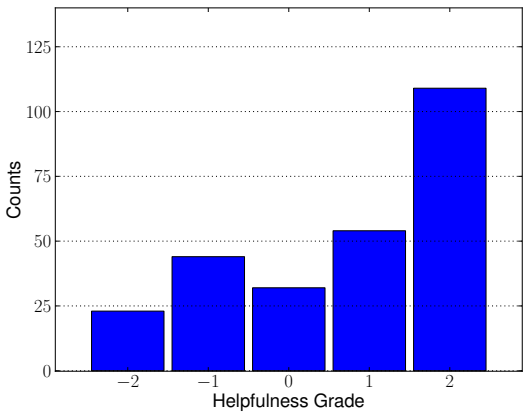

Figure 1: Histogram of review helpfulness grades for the Eng/Essay2 class.

Class	Average grade stdev
CS/Android	15.2%
CS/Web	10.4%
CS/C++	11.8%
CS/Java	14.5%
Eng/Essay1	8.0%
Eng/Essay2	8.2%
Econ	9.6%

Table 4: The average standard deviation of the grades received by individual submissions in a class, expressed as a percentage of the maximum grade M.

the only class in which review feedback and ratings were solicited uniformly: in the other classes, we had only a handful of ratings, compared to hundreds of reviews. As there is currently no incentive for students to rate and provide feedback on reviews, few students do so unless reminded by the instructor.

5.2 Quality of consensus grades

5.2.1 *Consistency of student grades*

Table 4 gives the average standard deviation of the grades received by individual submissions in a class. Precisely, for each submission i we compute $s_i = \text{std}_{j \in i \circ R}\{g_{ij}/M\}$, where $[0, M]$ is the grading range used for the assignment, and where $\text{std}\{\cdot\}$ is the standard deviation operator; we then report the average of s_i over all submissions i of a class.

In four CS/C++ assignments, we were also able to measure the difference between the consensus grades of submissions we knew were identical; this gives direct information on the "noise" present in CrowdGrader consensus grades. In these CS/C++ assignments, students were able to work in groups. Since at the time CrowdGrader did not support group submissions (the feature has since been added), the students were asked to each submit a solution: the student submissions would be graded independently, and the TA, who had a list of groups and their members, would later average the grades received by the students in the same group. This meant that we had available several pairs of identical submissions, coming from members of the same group, and graded by CrowdGrader independently one from the other. In Table 5, we report

Assignment	D	N. pairs
CS/C++ hw 2	18.0%	6
CS/C++ hw 3	11.8%	12
CS/C++ hw 4	10.3%	20
CS/C++ hw 5	10.9%	20

Table 5: Grade variation between independently graded identical submissions. D is the square root of the mean square difference of the grades received by identical submissions, expressed as a percentage of the maximum grade M.

the square root of the mean square difference $(\hat{g}_i - \hat{g}_l)^2$, computed over all pairs (i, l) of identical submissions, expressed as percentage of the grade range M. Except for CS/C++ Homework 2, where the number of identical submission pairs is only 6, we see that the difference is about 10%. As we will discuss in the conclusions, in this class there were some grading problems due differences in the development and testing environments used for the C++ code, and perhaps these problems contributed to the grade noise reported in Table 5.

5.2.2 *Evaluation using control grades*

For some assignments, we had available control grades given by the instructor, or other domain experts, for a randomly selected subset of submissions that numbered at least 20. We note that this evaluation is inherently approximate: while instructor and TAs are typically more knowledgeable than students in the subject matter, they can nevertheless make mistakes when grading homeworks, failing to spot problems, or not giving credit to great aspects of the work that go undetected.

For the Android assignments, the control grades were assigned by a teaching assistant who was a fairly accomplished Android developer. For the Java assignment, the control grades were provided by the instructor. For the CS/C++ assignments, the authors graded 20 or more randomly selected submissions for each assignment. We compared the control grades with the consensus grades according to the following metrics:

- ρ: the coefficient of statistical correlation (also known as Pearson's correlation) between the control grades $\{\tilde{g}_i\}$ and the consensus grades $\{\hat{g}_i\}$.
- norm-2: the norm-2 distance $(\sum_i (\tilde{g}_i - \hat{g}_i)^2)^{1/2}$ between the control grades $\{\tilde{g}_i\}$ and the consensus grades $\{\hat{g}_i\}$, expressed as percentage over the grading range M.
- s-score: we first normalize the control grades $\{\tilde{g}_i\}$ and the consensus grades $\{\hat{g}_i\}$, so that they both have zero mean and unit variance, obtaining $\{\tilde{g}'_i\}$, $\{\hat{g}'_i\}$ Then, we compute the standard deviation s of $\{\tilde{g}'_i - \hat{g}'_i\}$, and we report the s-score $1 - s/\sqrt{2}$.

The results for the various assignments are reported in Table 6. We see that, except for the Android Homework 3, the statistical correlation between consensus grades and control grades is about 0.8. For all of these homeworks, the difference between consensus and control grades averaged about 15%. The low correlation (and low s-score) for the Android Homework 3 can be explained by the fact that this was a particularly easy homework, in which many students received similar high grades. In other words, grades were concentrated in the high range of the scale $[0, M]$. So, even if the norm-2 distance of Android Homework 3 is similar to that of the other assignments, the correlation and norm-2 results are worse, owing to the more tightly clustered grades.

Assignment	ρ	norm-2	s-score
CS/C++ hw 2	0.75	14.0%	0.50
CS/C++ hw 3	0.84	14.9%	0.60
CS/Android hw 3	0.39	16.3%	0.22
CS/Java hw 2	0.85	17.5%	0.61

Table 6: Grading accuracy: consensus vs. control grades.

6. CONCLUSIONS

The main benefit students report from CrowdGrader consists in the quality of the feedback they receive, and in their ability to learn from studying the submissions of their classmates. From the point of view of the instructor, the benefit lies in facilitating student learning, and in the ability to handle the grading and evaluation of large classes. In this latter respect, the grading precision provided by CrowdGrader was sufficient for assigning, at the end of each class, the portion of the grade due to homework. If we assume that the control error has similar error to the consensus grade, and the errors are uncorrelated, the control error of about 15% reported in Table 6 corresponds to an error with respect to a "true" grade of $15\%/\sqrt{2} \approx 10\%$, which is roughly the noise in consensus grades measured in Tables 4 and 5. In our experience, such an imprecision is not uncommon in grading complex homeworks that require judgement on the part of the grader. Indeed, the number of students who complained about mis-gradings was about the same as the one we typically experience using TAs. When a mis-grading was reported, the instructor was able to read the reviews and the grades provided by the students, in addition to accessing the submission. Resolving the mis-gradings, when indeed they were mis-gradings, was simple; as usual, many of the reported mis-gradings were really mis-perceptions of the quality of one's own submission.

One of the authors taught the CS/Web and CS/Android classes. In these classes, we noted that students were generally more ready to reward originality than teaching assistants. The main goal of a teaching assistant is often to avoid controversy, in order to avoid confrontations with students. Thus, teaching assistants generally felt a stronger obligation to follow a rigid grading scheme, for the sake of consistency, and subtract a fixed number of points for each type of error encountered. Students felt less constrained by the need for full consistency, as the authors of the submissions they graded could not easily identify or compare the grades they received from the same grader.

In the computer-science assignments we considered, the greatest source of errors was the non-uniformity between the coding environments of the authors of homework submissions, and of the students who graded the submissions. In the C++ assignments, students would develop under linux, and their code was build under Mac OS X for grading, or vice versa: build instructions and Makefiles often did not work, or libraries were missing. In the Android assignments, students had to cope with the somewhat temperamental nature of the Android SDK in Eclipse, which occasionally malfunctioned, or with libraries that due to complex build settings were occasionally omitted. The fact that some students tested assignments on Android emulators, while others relied on actual phones or tablets, along with the different formats and screen sizes of the various Android devices, also contributed to evaluation variability. The clarity and precision of homework assignments is likely the major factor in the precision of any tool, or any TA, in evaluating submitted solutions.

Acknowledgements

We thank Ira Pohl at UC Santa Cruz for being an early adopter of CrowdGrader, and for providing insight and encouragement for this work. We thank Marco Faella at the University of Naples for agreeing to use Crowdgrader in his class when the tool was still in an early, very much experimental, version.

7. REFERENCES

[1] J. Bartholdi, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[2] D. J. Boud, R. Cohen, and J. Sampson. *Peer learning in higher education: learning from & with each other*. Psychology Press, 2001.

[3] R. Bradley and M. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):pp. 324–345, 1952.

[4] K. Cho, T. Chung, W. King, and C. Schunn. Peer-based computer-supported knowledge refinement: An empirical investigation. *Communications of the ACM*, 51(3):83–88, 2008.

[5] C. Davidson. How to crowdsource grading, 2009. `http://www.hastac.org/blogs/cathy-davidson/how-crowdsource-grading`.

[6] L. de Alfaro and M. Shavlovsky. Crowdgrader: Crowdsourcing the evaluation of homework assignments. Technical Report UCSC-SOE-13-11, UC Santa Cruz, 2013. arXiv:1308.5273.

[7] J.-C. de Borda. *Memoire sur les Elections au Scrutin*. 1781.

[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

[9] A. Elo. *The Rating of Chess Players Past and Present*. New York, Arco, 1978.

[10] M. Glickman. *Paired Comparison Models with Time-varying Parameters*. Harvard University, 1993.

[11] R. Luce. *Individual choice behavior : a theoretical analysis*. Wiley N.Y, 1959.

[12] M. Merrifield and D. Saari. Telescope time without tears: A distributed approach to peer review. *Astronomy & Geophysics*, 50(4):4–16, 2009.

[13] P. Naghizadeh and M. Liu. Incentives, quality, and risk: A look into the NSF proposal review pilot. *Arxiv*, 1307.6528v1, 2013.

[14] National Science Foundation. Dear colleague letter: Information to principal investigators (PIs) planning to submit proposals to the sensors and sensing systems (sss) program October 1 , 2013 deadline, 2013.