Pluggable Reputation Systems for Peer Review: a Web-Service Approach

Yang Song, Zhewei Hu, Edward F. Gehringer
Department of Computer Science
North Carolina State University
Raleigh, NC, U.S.
[ysong8, zhu6, efg]@ncsu.edu

***Abstract*—Peer review has long been used in education to provide students more timely feedback and allow them to learn from each other's work. In large courses and MOOCs, there is also interest in having students determine, or help determine, their classmates' grades. This requires a way to tell which peer reviewers' scores are credible. This can be done by comparing scores assigned by different reviewers with each other, and with scores that the instructor would have assigned. For this reason, several *reputation systems* have been designed; but until now, they have not been compared with each other, so we have no information about which performs best. To make the reputation algorithms pluggable for different peer-review system, we are carrying out a project to develop a reputation web service. This paper compares two reputation algorithms, each of which has two versions, and reports on our efforts to make them "pluggable," so they can easily be adopted by different peer-review systems. Toward this end, we have defined a Peer-Review Markup Language (PRML), which is a generic schema for data sharing among different peer-review systems.**

Keywords—educational peer review; reputation system; web service

I. INTRODUCTION

Online peer-review systems are now in common use in higher education. They free the instructor and course staff from having to provide personally all the feedback that students receive on their work [1]. However, if we want to assure that all students receive competent feedback, or even use peer-assigned grades, we need a way to judge which peer reviewers are most credible. The solution is to use a *reputation system* [2]. Reputation systems may take various factors into account:

- Does a reviewer assign scores that are similar to scores assigned by the instructor (on work that they both grade)?
- Does a reviewer assign scores that match those assigned by other reviewers?
- Does the reviewer assign different scores to different work?
- How competent has the reviewer been on other work done for the class?

Any or all of this information may be factored into a reviewer's reputation. Not only does this help us determine what grade to award student authors; it also helps students see how effective they are as reviewers.

This project is sponsored by the National Science Foundation, on grant DUE 1432347.

In the past, with small size classes, reputation systems were seen as an ancillary function of peer-review systems. However, in a large class they are essential, and in a MOOC, they are indispensable [3]. There is simply no other way to assess large volumes of work that cannot be automatically graded.

The same reputation algorithm can be used by many different peer-review systems. We have a multi-campus NSF-funded project to develop web services for peer review. Our project will develop pluggable reputation systems that can be deployed in many different peer-review systems. These reputation systems can assign grades calculated as a linear combination of student-assigned grades, with competent reviewers' scores weighted more heavily.

Once reputation systems have been deployed as web services, peer-review researchers will be able to use them to calculate scores on assignments, both past and present (past data can be used to tune the algorithms). This will yield the following benefits: (1) peer-review systems that lack reputation systems may use one of ours without coding it in their system; (2) instructors will be able to experiment with different reputation systems for their own classes, without implementing them locally; and (3) the information that we gather from this experimentation will help us recommend reputation systems appropriate for different kinds of assignments and courses.

II. ALGORITHMS SUPPORTED

Inputs of reputation algorithms vary, but a common way to represent review scores is to use a adjacency matrix. In this matrix, each row represents an artifact and each column represents a reviewer. The values of the matrix are the scores given to each artifact by each reviewer. Reputation algorithms use the matrix to generate two quantities: the reputation values for each reviewer and the weighted grades for each artifact.

It would be simplest to assign reputations based on agreement with expert (instructor) grades. But, in practice, instructor grades are not always available, so reputation algorithms need to calculate reviewer reputations and weighted scores for artifacts recursively till they reach convergence.

To give a precise definition of the algorithms supported in our web service, let a be an artifact; A be the set of all the artifacts; r be a reviewer; R be the set of all the reviewers; g_a^r

978-1-4799-8454-1/15/$31.00 ©2015 IEEE

be the grade that r assigned to a; R_r be the set of artifacts reviewed by r; A_a be the set of reviewers who have reviewed a; W_r be the weight for reviewer r; G^a_{expert} be the expert grade for artifact a; $G^a_{predict}$ be the grade that the algorithm predicts for artifact a based on current reputations; and G^a be the temporal grade for artifact a in the algorithm.

In the Hamer algorithm (we call it Hamer-peer in this paper) [4], $G^a_{predict}$ is the weighted average of the grades assigned to a:

$$G^a_{predict} = \sum_{r \in A_a} g^r_a \times W_r \Big/ \sum_{r \in A_a} W_r \tag{1}$$

$G^a_{predict}$ is then used to update the reviewers' weights since the expert grades are not considered available: $G^a = G^a_{predict}$.

The variances in the scores assigned by each reviewer are calculated based on how close their grades are to the predicted weighted grades:

$$\Delta_r = \sum_{a \in R_r} (G^a - g^r_a)^2 \Big/ |R_r| \tag{2}$$

With the Δ_r for all the $r \in R$, we can tell whether a reviewer's variance is higher or lower than average:

$$W_r' = mean\Delta_r / \Delta_r \tag{3}$$

So the higher W_r' is for $r \in R$, the "better" a reviewer r is. The range for W_r' is $(0, \infty)$.

To avoid having the reviewers with higher W_r' "dominate" the grade aggregation, Hamer used a "log-damping" approach to calculate the W_r:

$$W_r = \begin{cases} 2 + \log(W_r' - 1) & W_r' > 2 \\ W_r' & W_r' \le 2 \end{cases} \tag{4}$$

W_r and $G^a_{predict}$ are calculated iteratively till they converge.

However, if expert grades are available, it is unnecessary to calculate $G^a_{predict}$; instead, G^a_{expert} can be used as G^a: $G^a = G^a_{expert}$. With this modification, we can use similar processes to derive an implementation of the Hamer-peer algorithm with expert grades (Hamer-expert). The difference is that this algorithm does not need to be run multiple times because G^a does not change from one round to the next.

The ranges for both Hamer-peer and Hamer-expert are $(0, \infty)$. If a review's reputation score is greater than 1, this reviewer is considered to be above average.

In contrast to the Hamer algorithms, the reputation range calculated by the Lauw algorithm (we call it Lauw-peer in this paper) [5] is $[0,1]$. A reputation score close to 1 means the reviewer is credible.

The main difference between the Hamer-peer and the Lauw-peer algorithm is that the Lauw-peer algorithm keeps track of the reviewer's leniency ("bias"), which can be either positive or negative. A positive leniency indicates the reviewer tends to give higher scores than average.

Let l_r be the leniency of r. After $G^a_{predict}$ is calculated for each $a \in A$, l_r can be updated:

$$l_r = \sum_{a \in R_r} (G^a - g^r_a) / g^r_a \Big/ |R_r| \tag{5}$$

The difference between (2) and (5) is that in the Lauw-peer algorithm, the variance is not squared, which results in higher reputation scores for the weak reviewer. More results and analysis are provided in section 4.

Similar to the Hamer-expert algorithm, if expert grades are available, a new algorithm (Lauw-expert) can be defined by using G^a_{expert} as G^a. This algorithm also does not need to be run recursively because expert grades are already available.

III. Design of the Web Service and PRML

A. Design of reputation web service

Our web service is designed to make different reputation algorithms available for peer-review systems. As in any web service, the inputs and outputs will follow a pre-defined format. Our design goals were to—

- minimize data transactions;
- minimize the effort needed to plug the web service into the clients;
- minimize the knowledge that the web services need to have about the clients.

Fortunately, no personal information needs to be passed to the web service; personal IDs and names are not needed for calculating reputations. The web service can keep track of each reviewer using the primary keys from the clients, or a one-way hash, for example. Nor does the web service need to store any information used to calculate the reputations. However, client systems may choose to allow it to save anonymized score information to, for example, calculate reputations for a whole semester, or to make it available in a learning-analytic database.

To make use of all the algorithms provided by our web service, client systems simply need to implement a web-service interface module for their systems. This module will (1) transform and send data to the web service and (2) receive the reputation data and use it in calculating grades. We are open to helping potential clients to implement this module. Since this interface is on the clients' end, clients can set up security rules to guarantee data integrity and security. The clients will be able to adapt this web service interface module to make use of our other web services (such as visualization and rubric improvement).

The data that the web service receives from the pluggable interfaces will be in a common format, regardless of how the client systems represent it. A common format also simplifies the design of reputation web services, because they do not need to have any knowledge of the development languages, the designs and the purposes of the client systems.

B. *Design of PRML*

In order to interface this reputation web service to a wide variety of peer-review systems, we have defined a Peer-Review Markup Language (PRML). This language will provide a standard format for representing data structures common to peer-review systems, such as assignments, users, rubrics, and reviews. It will allow different peer-review systems to interface to our reputation web service (and our other web services) without needing to change their databases.

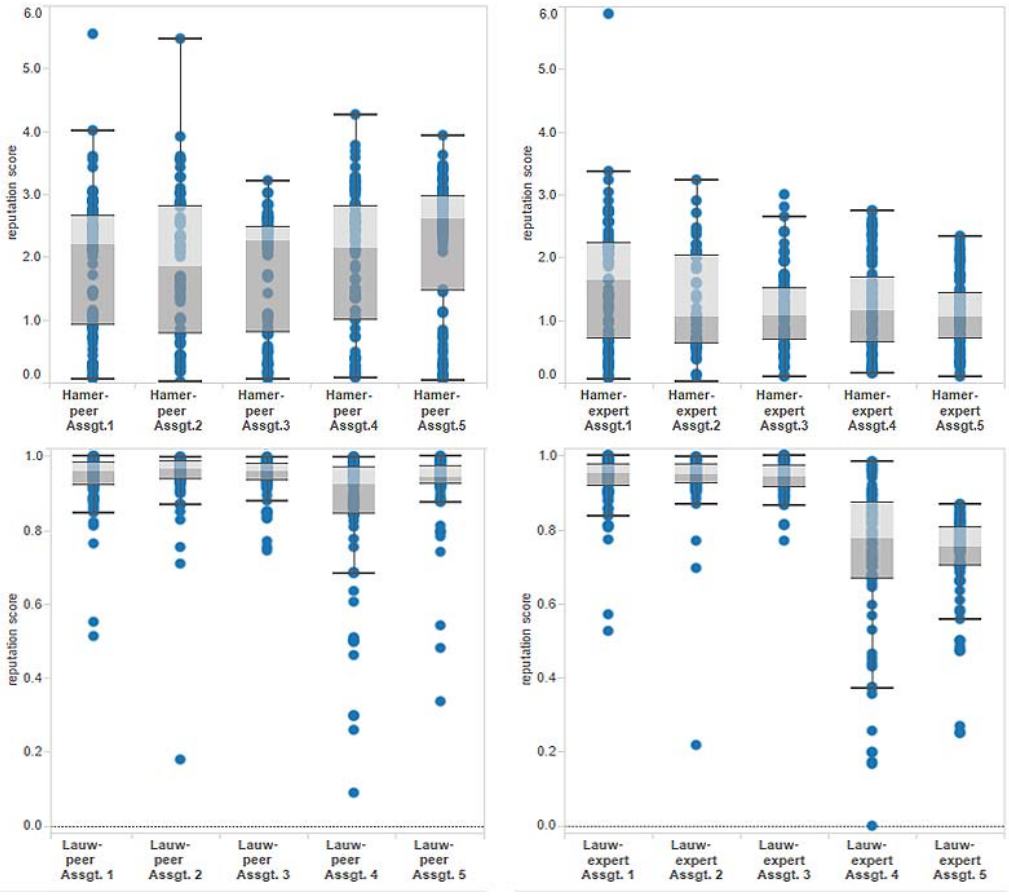

Fig. 1. Distribution of reputation scores caculated by each algorithm.

PRML is a generic data schema for collecting and sharing numeric and textual data. The reputation web service uses only a subset of PRML. The full PRML contains other entities, such as textual feedback provided by students. This paper only presents the portion of PRML used for reputation services.

PRML is an JSON-based markup language which encapsulates data and metadata generated by peer-review tasks. The entities of data used in the reputation web service are clients, assignments, tasks, reviewers, reviewed entities and scores. Each of these is explained in Table I. [a]

TABLE I. ENTITIES NEED BY THE REPUATION WEB SERVICE

Entity name	Explanation
Client	A client system that uses the reputation web service.
Assignment	An assignment is what an instructor creates to prompt students to create artifacts.
Task	A task is an activity within an assignment. To incorporate peer-review systems that support multiple rounds of reviews, we consider each round as a task, and the web service calculates repuations for each task instead of for each assignment.
Reviewer	A user who submits one or more reviews for the current task.
Reviewed entity	The reviewed entity in the current task can be an essay, a video, a program, etc. Since we do not do content analysis in the reputation web service, we only need identifiers for the entities.
Score	The mapping table between reviewers and revieweed entities. Each instance represents a peer-review record, consisting of {reviewer, revieweed_entities, task, score assigned}. The instances of this entity should also contain the expert grades if they are available.

IV. DATA SET AND EXPERIMENTAL RESULTS

The data set that we used in experiments was generated in CSC 517 (Object-Oriented Design and Development) at NC State University in 2014. This is a graduate-level course, which peer-reviews two programming and three writing assignments. All of these are team assignments with 2- to 4-member teams. In the review phase, students select artifacts to review. Peer reviews are done using the Expertiza system [6], which is a system developed to review student-generated course content. In the peer-review phases for each assignment, reviewers were asked to fill out a review rubric. They were expected to give textual feedback and Likert-scale scores for the various rubric criteria.

a. Class diagram of these categories can be found in : http://www4.ncsu.edu/~ysong8/FIE2015/class%20diagram.jpg. Example of PRML can be found in: http://www4.ncsu.edu/~ysong8/FIE2015/json%20sample.html.

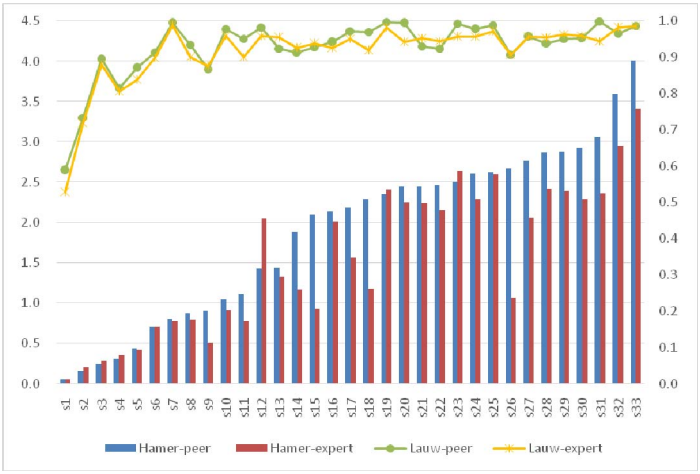

Fig. 2. Distribution of reputation scores caculated by each algorithm. We sampled 33 students (s1–s33) from assignment 1. The barchart shows the reputations calculated by Hamer-peer and Hamer-expert algorithms, with scale on the left side; the line chart shows the reputations calculated by Lauw-peer and Lauw-expert algorithm, with scale on the right.

Eighty-four students finished the course, and 2852 peer reviews were collected. We used the peer-review scores and computed aggregated scores on a 0-to-100 scale. Some of the artifacts received more than 100 because of bonus points, but bonus points are ignored in our experiments.

The web-service code calculated reputations for all students. Fig. 1. shows the distribution of all reviewers' reputation for all the assignments. The distribution of the reputation scores show that the Hamer algorithms have wider ranges, and that weak reviewers can receive reputation scores which are close to 0. By contrast, the Lauw algorithms give decent reputation scores to most of the students, but accord less weight to peer grades from credible reviewers.

One question that may interest instructors on client systems is, Do the reputation algorithms give consistent results? To answer this question we sampled 1/3 of students from assignment 1. Their reputations are presented in Fig. 2. It shows that even though the ranges of those algorithms vary, the result are consistent: the reviewers found to be credible by one algorithm are also deemed credible by the other algorithms. This is because all the algorithms are based on the degree of consistency between the scores given by one reviewer versus the scores given by the other reviewers on the same artifact. Another observation from Fig. 2 is that the students who received reputation scores close to 1 from the Hamer algorithms (meaning peer reviews skills are average) may receive 90% reputation scores from both Lauw algorithms, which shows that Lauw's tends to give higher reputation scores.

Another piece of information that the instructors using the client systems may want to know is, Which algorithm works best for my course? Since in this case the experts grades are available, the client can calculate the cosine values between the predicted weighted grades and expert grades for each assignment. We sample three assignments from the data set, namely assgt. 1 (writing a wiki page) assgt. 3 (writing a design document) and assgt. 5 (programming). We also computed the overall bias and maximum bias of predicted grades, relative to expert grades. The result is shown in Table II.

TABLE II. COSINES BETWEEN PREDICTED AND EXPERT GRADES

		Hamer-peer	Hamer-expert	Lauw-peer	Lauw-expert
Assgt. 1	***Cosine***	0.999	0.999	0.999	0.999
	Overall bias	1.627	–0.773	1.082	–0.755
	Max. absolute bias	7.9	12.2	15.6	12.1
Assgt. 3	***Cosine***	0.997	0.997	0.997	0.997
	Overall bias	4.824	3.210	2.533	2.405
	Max. absolute bias	23.8	22.5	22.6	22.5
Assgt. 5	***Cosine***	0.999	0.999	0.998	0.998
	Overall bias	–16.583	–16.489	–18.539	–18.022
	Max. absolute bias	23.2	23.5	26.9	26.3

All the algorithms perform well on assignment 1: the Hamer-peer algorithm has the lowest maximum absolute bias and the Lauw-peer algorithm has the lowest overall bias. This indicates, from the instructor's perspective, if there are further assignments of this kind, expert grading may not be necessary. From assgt. 3, the overall bias is a little bit higher, but the max. absolute bias is very high (more than 20). This indicates that for future similar courses, the instructor can trust most students' peer grading, but should be aware that the students may give inflated grades. Therefore spot-checking is necessary. For assgt. 5, however, overall bias is quite low, as the students gave grades at least 16 points lower than expert grades. This may because either more training is needed, or the review rubric is inadequate. The results on assgt. 5 also suggest that for future courses of this kind, the instructor cannot trust the students' grades; expert grades are still necessary.

V. CONCLUSION

Reputation systems are a necessary part of peer-review systems, especially when they are used by large classes. We have designed a reputation web service which makes reputation systems "pluggable" into any peer-review system. It provides several algorithms that allow client system users to test different reputation algorithms without the need of implementing them locally. Instructors using the client systems can use reputation scores as weights in aggregating student-assigned scores into grades for each artifact.

The reputation web service is based on our Peer-Review Markup Language (PRML), which is designed as a generic schema for sharing peer-review based educational data. PRML isn't just useful for communicating with web services; it also facilitates development of more advanced peer-review systems by giving the community a common language for talking about peer review. With the plug-in interfaces that support PRML, researchers can also (1) share data with each other, and (2) build a centralized data repository. This means that, PRML can serve as a bridge to bring together isolated peer-review systems into a single research community.

REFERENCES

[1] K. Topping and S. Ehly, *Peer-assisted Learning*. Routledge, 1998.

[2] E. F. Gehringer, "A Survey of Methods for Improving Review Quality," in *New Horizons in Web Based Learning*, Y. Cao, T. Väljataga, J. K. T. Tang, H. Leung, and M. Laanpere, Eds. Springer International Publishing, 2014, pp. 92–97.

[3] C. Piech, J. Huang, Z. Chen, C. Do, A. Ng, and D. Koller, "Tuned Models of Peer Assessment in MOOCs," *ArXiv13072579 Cs Stat*, Jul. 2013.

[4] J. Hamer, K. T. K. Ma, and H. H. F. Kwong, "A Method of Automatic Grade Calibration in Peer Assessment," in *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, Darlinghurst, Australia, Australia, 2005, pp. 67–72.

[5] H. W. Lauw, E. Lim, and K. Wang, "Summarizing review scores of 'unequal' reviewers," in *In Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[6] E. Gehringer, L. Ehresman, S. G. Conger, and P. Wagle, *Reusable Learning Objects Through Peer Review: The Expertiza Approach*. .