Peer Assessment and Knowledge Discovering in a Community of Learners

Maria De Marsico[1], Filippo Sciarrone[2], Andrea Sterbini[1] and Marco Temperini[2]
[1]*Dept. of Computer Science, Sapienza University,*
Via Salaria, 113, 00189 Roma, Italy
[2]*Dept. of Computer, Control and Management Engineering, Sapienza University,*
Via Ariosto, 25, 00184 Roma, Italy

Keywords: Peer Assessment, Machine Learning, Student Modeling.

Abstract: Thanks to the exponential growth of the Internet, Distance Education is becoming more and more strategic in many fields of daily life. Its main advantage is that students can learn through appropriate web platforms that allow them to take advantage of multimedia and interactive teaching materials, without constraints neither of time nor of space. Today, in fact, the Internet offers many platforms suitable for this purpose, such as Moodle, ATutor and others. Coursera is another example of a platform that offers different courses to thousands of enrolled students. This approach to learning is, however, posing new problems such as that of the assessment of the learning status of the learner in the case where there were thousands of students following a course, as is in Massive On-line Courses (MOOC). The Peer Assessment can therefore be a solution to this problem: evaluation takes place between peers, creating a dynamic in the community of learners that evolves autonomously. In this article, we present a first step towards this direction through a peer assessment mechanism led by the teacher who intervenes by evaluating a very small part of the students. Through a mechanism based on machine learning, and in particular on a modified form of K-NN, given the teacher's grades, the system should converge towards an evaluation that is as similar as possible to the one that the teacher would have given. An experiment is presented with encouraging results.

1 INTRODUCTION

Thanks to the exponential growth of the Internet that has occurred in recent years, many fields have changed or are radically changing their approach to training. Today many distance courses are offered on the web, such as *Coursera* [1] and *Khan Academy* [2], provided through appropriate technology platforms available 24 hours a day. This approach is proving a great success for various and obvious reasons: first of all, the user can manage his training time, without any space or time restrictions. Moreover, with the advent of *HTML5*, the available teaching materials can present strong multimedia and interactive features, making learning even more enjoyable. Another important aspect is that of communities of learning where professionals, but also common people, propose learning paths. The number of participants must also be taken into consideration: a specific university course can have 200 students using a platform while courses like those proposed by Coursera can boast thousands. These new scenarios pose new aspects and problems: modern pedagogy is re-evaluating the theory of social constructivism in which students also learn through peer interactions (Vygotsky, 1962), while the aspect of student assessment, with big numbers, requires a re-thinking of the approach. It would be impossible for a teacher to correct thousands of assignments. For this reason, in recent years software tools are being developed for the automatic correction of open answers assignments. On the other hand, it is not always possible to monitor progress in a learning path by means of summative assessments by closed answers (such as tests). The work that we present in this article deals with this last aspect: a novel semi-automatic method that helps the teacher to evaluate a community of students for open answers assignments. In other articles, we have already addressed this problem with the OpenAnswer system, where a mechanism of correction of open-ended questions was proposed, with

[1]https://www.coursera.org
[2]https://it.khanacademy.org/

Marsico, M., Sciarrone, F., Sterbini, A. and Temperini, M.
Peer Assessment and Knowledge Discovering in a Community of Learners.
DOI: 10.5220/0007229401190126
In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 1: KDIR*, pages 119-126
ISBN: 978-989-758-330-8

the support of the teacher. The mechanism was based on Bayesian Networks (De Marsico et al., 2017a) while in (De Marsico et al., 2017b) a first version of a modified K-NN technique was presented. Here we face the same problem but with different variations in the learning algorithms and in the student models. First of all we enhance the Student Model (SM), adding another stochastic variable, the *Dev* variable, representing the credibility of the *Knowledge Level K*. Furthermore we propose a more complete version of the learning algorithms representing a modified version of K-NN (Mitchell, 1997). Finally, a novel simulation environment is used in order to simulate communities of learners. In Section 2 we present a brief review of the literature relevant to the work; in Section 3 the algorithms are shown. In the Section 4 we illustrate an experimental evaluation in a simulated environment and finally in the Section 5 the conclusions and future developments are drown.

2 RELATED WORK

The literature offers many articles proposing Machine Learning techniques and, more generally, Artificial Intelligence algorithms for the study of the dynamics both of individuals and of communities of students (Limongelli et al., 2008; Limongelli et al., 2013; Limongelli et al., 2015). Here we address some works worth of mention for peer-assessment.

Peer-assessment (Kane and Lawler, 1978) is an activity in which a student (or a group) is allowed to evaluate other students assignments (and possibly self-evaluate own assignments). It can be organized in different ways, yet a basic aspect is that it can be considered as one of the activities in which social interaction and collaboration among students can be triggered. It can also serve as a way to verify how the teacher can communicate to the students her own quality requirements with respect to the learning topics: if this happens, assessments from peers and from teacher agree better (Sadler and Good, 2006).

Student involvement in assessment typically takes the form of peer assessment or self assessment. In both of these activities, students are engaging with criteria and standards, and applying them to make judgments. In self assessment, students judge their own work, while in peer assessment they judge the work of their peers (Falchikov and Goldfinch, 2000).

Peer assessment is grounded in philosophies of active learning (Piaget, 1971) and androgogy (Cross, 1981), and may also be seen as being a manifestation of social constructionism (Vygotsky, 1962), as it often involves the joint construction of knowledge through discourse.

A peer assessment system to be mentioned is the proposal of (De Marsico et al., 2017a) where the *OpenAnswer* peer assessment system is presented. A peer assessment engine, based on Bayesian networks, is trained for the evaluation of open ended questions. The system is based on the SM, composed by some stochastic variables, such as the variable *K* representing the learner's *Knowledge Level*, and the variable J representing the learner's ability to judge the answers of her peers. Students initially grade *n* open-ended exercises of their peers. Subsequently, the teacher grades *m* students. Each student has therefore associated a *Conditional Probability Table* that evolves with time. This system has the same goal of our system but is based on different mechanisms. The Bayesian network presents some aspects of complexity that make the whole system a black box and little treatable for large numbers of students, as in the case of MOOCs, while our learning system has a much lower complexity and does not present problems of intractability. Another work (Anson and Goodman, 2014) proposes peer assessment to improve Student Team Experiences. An online peer assessment system and team improvement process was developed based on three design criteria: efficient administration of the assessment, promotion of quality feedback, and fostering effective team processes. In (Sterbini and Temperini, 2012) the authors propose an approach to open answers grading, based on Constraint Logic Programming (CLP) and peer assessment, where students are modeled as triples of finite domain variables. The CLP Prolog module supported the generation of hypotheses of correctness for answers (grounded on students peer-evaluation), and the assessment of such hypotheses (also based on the answers already graded by the teacher).

3 THE PEER ASSESSMENT ENGINE

In this Section we show the algorithms and the rationale of our proposal. Here we present an enhanced version of the engine presented in (De Marsico et al., 2017b). The most important differences are: the generation of a simulating environment producing the sample and different student model evolution taking into account some community aspects. The inference engine is based on a *learning* algorithm: K-NN. This is a *Lazy Learning* approach (e.g. (Mitchell, 1997)), also referred to as *Instance Based* learning: basically, in order to *learn* better classifying elements, the algorithm adapts the classification to each further instance

of the elements, that becomes part of the training set. Each training instance is represented as a point in the n-dimensional space of the instance attributes.

3.1 The Student Model

Each student is represented by a Student Model (SM), $SM \equiv \{K, J, Dev, St\}$, composed by the following variables:

- $K \equiv [1,10]$. Practically, it is the grade that the teacher has assigned to her through the correction of one or more structured open-ended exercises. From a learning point of view, it represents the learner's competence (*Knowledge level*) about the question domain
- $J \equiv [0,1]$. It is a measure of the learner's assessing capability (*Judgement*) and depends on K.
- Standard Deviation *Dev*. it represents the credibility of the value of K. The higher this value, the less the value of K of the student is credible. *Dev* is calculated as the standard deviation generated, for each i-th learner as follows:

$$Dev_i = \sqrt{\frac{\sum_{l=1}^{n}(K_i - K_l)^2}{n}} \quad (1)$$

being each K_l one of the group of students that graded her;

- $St \equiv \{CORE, NO\_CORE\}$. Each student can be in two different states: CORE and NO_CORE. Initially all the students are NO_CORE. If the student is voted by the teacher then she becomes a CORE student. These students, as we will see later, are important for the dynamics of the network. Each NO_CORE student is represented as s^- while a CORE student is represented as s^+. Consequently, the community of students is, at any given moment, dynamically parted into two groups: the *Core Group* (CG), and its complement $\overline{CG}$. CG is composed by the students whose answers have been graded directly by the teacher: for them K is given (fixed). In the following we also call this set as S^+, and call its elements the s^+ students. On the contrary, S^- is the set of students whose grade is to be inferred (so, they have been graded only by peers).

By this SM representation, each learner can be represented as a point in a 2-dimensional space (K,J).

3.2 Student's Model Initialization

First the each SM is initialized as follows:

- The teacher assigns an open-ended question to all the students;
- Each student provides an answer;
- Each student grades the answers of n different peers, and her answer receives n peer grades;
- each s_l^- student model, $SM_l = \{K_l, J_l, Dev_l, St_l\}$, is initialized as follows:

$$K_l^- = \frac{\sum_{i=1}^{n} K_i^-}{n} \quad (2)$$

where K_i^- is the grade received by the $i-th$ of the n peers who graded the s_l^- student. In this way, the K_l^- value is initialized with the mean of all received grades. The rationale is that in this step we do not know the differences among students' true assessment capabilities, and so we give to each of them the same weight.

For each s_l^- student, J_l^- is initialized as follows:

$$J_l^- = \frac{1}{1 + \sqrt{\sum_{i=1}^{n} \Delta_i{}^2}} \quad (3)$$

$\Delta_i^2 = (K_{lj} - \overline{K_j})^2$, being K_{lj} the grade assigned by the student s_l to the student s_j and $\overline{K_j}$ the arithmetic mean, i.e., the initial K^- of the student s_j, computed by Eq. 2.

So, if a student grades her n peers with values always equal to their K^- values, her J^- value gets maximal: $J = 1$ (here we haven't teacher's grades available, so we have to do with the peer evaluations only).

- All students are initialized to $St = NO\_CORE$;

The above mentioned elements are of course students, which we represent by a two-variables Student Model (SM): $K \equiv [1,10]$ and $J \equiv [0,1]$. K represents her competence (*Knowledge level*) about the question domain; J is a measure of her assessing capability (*Judgement*). By such attributes, each student is in turn represented as a point in the (K,J) space.

Once the whole peer-evaluation has been completed, and no teacher's grading has yet been performed, our module's overall learning process starts with an initialization step: the students' SMs are initialized basing solely on the peer-evaluation data. Then, the learning process continues: at each following step, some answers from the S^- students are graded by the teacher, and consequently some students are extracted from S^- and added to S^+, and the *SMs* are recomputed: in particular, at each step the positions of the points representing S^- students, in the (K,J) space, do change, implying a new classification for them, which depends on their distance from points in S^+, according to the K-NN protocol.

At each step the module *learns* to (hopefully) better classifying the students in S^-, until a termination

condition suggests to stop cycling, and the S^- students *SMs* become the grades finally inferred by the module.

3.3 Student's Model Evolution

After the SM initialization, all learners belong to the S^- set. Each learner evolves in the (K,J) space as follows:

- The teacher is suggested a ranked list of students/answers to grade, sorted by the *Dev* key. The variable *Dev*, for each learner is a very important variable because it represents the difference between the knowledge level K and at a certain moment and how much this differs from the individual evaluations given by the peers to the student himself. A very high *Dev* means having a K that is not very believable as the student has received from his n peers ratings very different from each other and then in this case a teacher's intervention on the value of K could be very positive.
- The teacher selects a group of students/answers in the ranked list, and grades them. Such grades are the new, final, K^+ values for such students;
- The graded students become s^+ students, and their position in the (K,J) space changes;
- A chain all peers who had voted for the student who became s^+ change their model. The model updating algorithm follows recursively a graph path starting from the voted students and so on backwards. For each learner, first K, and J are updated. Once all the students influenced by the teacher's vote have been updated, all their *Dev* updated.

In the following we will use *KMIN* and *KMAX* to denote the minimum and maximum values for K (i.e. here respectively 1 and 10). *IMAX* will denote the maximum difference between two values of K, i.e. here 9. Moreover *JMIN* and *JMAX* will denote the minimum and maximum values for J (i.e. here resp. 0 and 1). Finally, Dev_{min} and Dev_{max} represent the lowest and highest values for the variable *Dev*, i.e., $DevMIN = 0$ and $DevMAX = 9$.

The SM updating is explained in detail in the next paragraphs.

3.3.1 Updating of the Graded Learner

The graded learner SM is updated. First the K value is updated:

$$K^+ = K_{teacher} \tag{4}$$

being $K_{teacher}$ the grade assigned by the teacher. Secondly the J value:

$$\begin{aligned} J^+_{new} &= J_{old} + \alpha(JMAX - J_{old}) \ \ (0 \leq \alpha \leq 1) \\ J^+_{new} &= J_{old} + \alpha J_{old} \ \ (\alpha < 0) \\ \alpha &= \frac{K_{teacher} - K^-_{old}}{IMAX} \end{aligned} \tag{5}$$

Notice, in Eq.5:

1. A convex function has been adopted for J update, providing the two cases according to the possible value of α. In particular J_{old} could stand for J^+_{old} or J^-_{old}, depending on the student being already in S^+ (case J^+_{old}), or being just entering in S^+ (case J^-_{old}) or remaining in S^- (case J^-_{old} again).
2. In general we assume that the assessment skill of a student depends on her *Knowledge Level K*, so the J value is a function of K. In the case $K_{teacher} = K^-_{old}$, no change is implied for J. Also notice that the difference $K_{teacher} - K^-_{old}$ is normalized with respect to I_{max}. If the student receives a grade higher than her current one, we increase her *Judgement Level*: the higher the level of knowledge, the higher is her judgment capability. Otherwise J decreases. Equation 5 increases or decreases J by an amount such that its value always remains in the range $[0,1]$. Moreover, we used this this type of evolutionary form as it is the easiest to treat as a first approach and also because it is used very often in automatic learning as an update of statistical variables in a machine learning context (see for example (Bishop, 2006)).

Subsequently the value of *Dev* is modified recalculating it on the student voted by the teacher, then according to the same rule used, that is the equation 1, i.e.:

$$Dev_{new} = \sqrt{\frac{\sum_{l=1}^{n}(K_{teacher} - K_l)^2}{n}} \tag{6}$$

3.3.2 Other SMs Updating

Once the student who has been voted by the teacher has changed his model, consequently the algorithm recursively changes the models of all the students. The students community, from the point of view of the data structure that represents it, can be seen as a weighted oriented graph where each node is a student and the following rules apply:

- Two nodes s_i and s_j are connected by a weighed edge iff s_i graded s_j ($s_i \longrightarrow s_j$) or s_j graded s_i ($s_j \longrightarrow s_i$);
- each edge is tagged with a weight w_{ij}, representing the grade that the student s_i gave s_j;

In Fig. 1 an example of the graph. So the algorithm recursively works on the adjacency matrix starting from graded student. For each student (i.e. a node), not a CORE student, the algorithm modifies the SM. All the students, s^-, who are influenced by the graded student are modified, according to the following rules (students s^+ are fixed because graded by the teacher):

$$\begin{aligned} K^-_{new} &= K^-_{grading} + \alpha(KMAX - K^-_{graded}) \ \ (0 \leq \alpha \leq 1) \\ K^-_{new} &= K^-_{graded} + \alpha K^-_{graded} \ \ (\alpha < 0) \\ \alpha &= \frac{1}{IMAX}(K^-_{grading} - K^-_{graded})\frac{Dev_{grading}}{IMAX} \end{aligned} \tag{7}$$

where: K_{new} is the new value of K of the intermediate student (in Fig. 1it is the $s1$ node). The $\frac{Dev_{grading}}{IMAX}$ factor expresses a kind of inertia of the value of K to change: the higher this value is, the more the value of K changes. The rationale behind this choice is that a student with a value of his own *Dev* high is a student who has received very different grades from those peers who graded her and therefore is better that it changes. each J value is changed as follows:

$$\begin{aligned} J^-_{new} &= J^-_{grading} + \beta(JMAX - J^-_{grading}) \ \ (0 \leq \beta \leq 1) \\ J^-_{new} &= J^-_{grading} + \beta J^-_{grading} \ \ (\beta < 0) \\ J^-_{new} &= J^-_{grading} + (K^-_{grading} - K^-_{graded}) \\ &\quad (\beta = 0 \wedge J^-_{grading} = J^-_{graded}) \\ &\quad with: \\ \beta &= \frac{1}{IMAX}(K^-_{new} - K^-_{grading})|J_{grading} - J_{graded}|\frac{Dev_{grading}}{IMAX} \end{aligned} \tag{8}$$

After, in order to complete the SMs, all the *Dev* variables are updated.

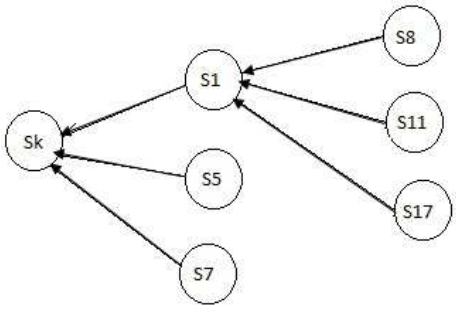

Figure 1: An extract of the graph. The teacher has voted for the student s_k. Starting from this student, the algorithm dates back to changing the models of the students who voted for it. First s_1, then s_8, s_{11}, s_{17}. Then it goes to s_5 and s_7.

3.4 K-NN Network Evolution

Finally, after that the teacher has graded some s^- students, become s^+ students, the modified K-NN algorithm can start. The learning process, is composed by the following equations:

$$\begin{aligned} K^-_{new} &= K^-_{old} + \alpha(KMAX - K^-_{old}) \ \ (0 \leq \alpha \leq 1) \\ K^-_{new} &= K^-_{old} + \alpha(1 - K^-_{old}) \ \ (\alpha < 0) \\ \alpha &= \frac{1}{I_{max}} \frac{\sum_{i=1}^{k} \frac{1}{d_i}(K^+_i - K^-_{old})}{\sum_{i=1}^{k} \frac{1}{d_i}} \frac{Dev_i}{IMAX} \end{aligned} \tag{9}$$

where:

1. d_i is the Euclidean distance between the s^-_{old} student under update, and the $i-th$ student in the Core Group (s^+_i);
2. The K^-_{new} value is given as a convex function, to keep K in $[1,10]$;
3. the acronym K-NN features a K, possibly misleading here, so we are using k for the number of nearest neighbors to be used in the learning algorithm.
4. The $\frac{Dev_i}{IMAX}$ factor has the same meaning of....

$$\begin{aligned} J^-_{new} &= J^-_{old} + \frac{(K^-_{new} - K^-_{old})}{IMAX} J^-_{old} \ \ (\beta = 0 \ \wedge \ J^+_i = J^-_{old}, i = 1 \ldots k) \\ J^-_{new} &= J^-_{old} + \beta(JMAX - J^-_{old}) \ \ (0 \leq \beta \leq 1) \\ J^-_{new} &= J^-_{old} + \beta J^-_{old} \ \ (\beta < 0) \\ \text{with } \beta &= \frac{(K^-_{new} - K_{old})}{IMAX} \frac{\sum_{i=1}^{k} \frac{1}{d_i}|J^+_i - J^-_{old}|}{\sum_{i=1}^{k} \frac{1}{d_i}} \frac{Dev_i}{IMAX}. \end{aligned} \tag{10}$$

where:

1. As mentioned earlier, we assume J depending on K: this is expressed through the difference between the K^-_{new} value, obtained by Equation 9, and the K^-_{old} value.
2. d_i is the Euclidean distance between the s^-_{old} student under update, and the $i-th$ student in the Core Group (s^+_i);
3. The J_{new} value is given as a convex function, to keep J in its normal range $[0,1]$;
4. k is as explained in the previous equation.
5. About the coefficient β, some notices are due, for the cases when $\beta = 0$. On the one hand, when the J^+ of the k nearest neighbors is equal to the J^-_{old} value of the s^-_i student under update, J^-_{new} is computed by the difference between K^-_{new} and K^-_{old} only. The rationale is that when the s^- student changes her K^- value, her assessment skill

should change as well (by the assumption of dependence of J on K). On the other hand, when the K^- value for the student under update is not changed, the assessment skill stays unchanged as well.

4 EXPERIMENTAL EVALUATION

In this Section we show an experimental evaluation of the algorithms for the network dynamic. The goal of this evaluation is to check the validity of the proposed algorithms, i.e., to show that, after some grades that as the teacher directly votes the students, the network modifies their models so as to converge all towards the votes that would have given the teacher, obviously with a certain gap. We built a software system where to run our trials. In this way, a teacher should not correct all the assignments but only a part of them, consuming less time.

The evaluation of such a system presents various problems related to the sample of users as the proposed algorithms have been designed to address community of students also formed by large numbers as in the MOOC jar where there may be courses with hundreds or even thousands of students. So, for a first experimentation we created an environment that generates sets of students from well-known and realistic statistical distributions for the sector. For the grades assigned by the teacher to the students we referred to a Gaussian distribution, generated with the statistical environment R, while regarding the simulation of the initial models of the students we referred to a uniform distribution of the votes assigned among peers.

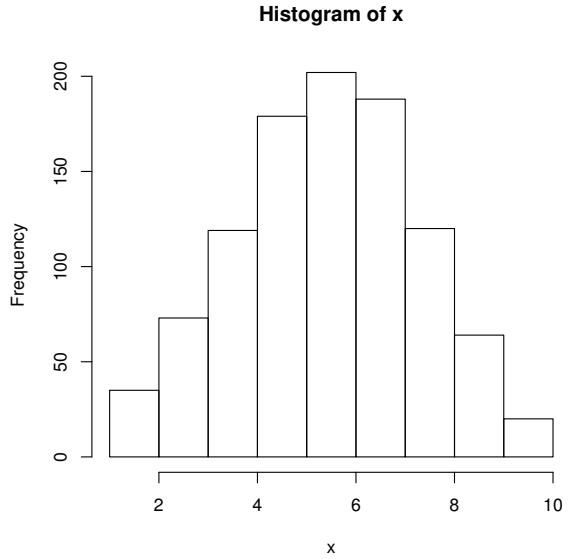

Figure 2: The distribution of the teacher grades for n=1000 students.

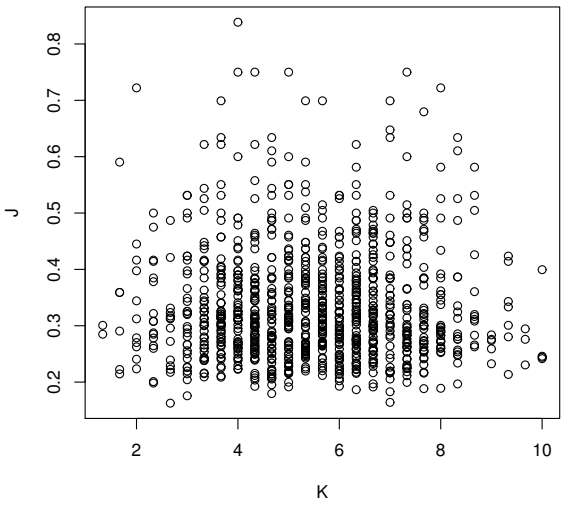

Figure 3: The distribution of the initial n=1000 SMs where each student graded 3 peers.

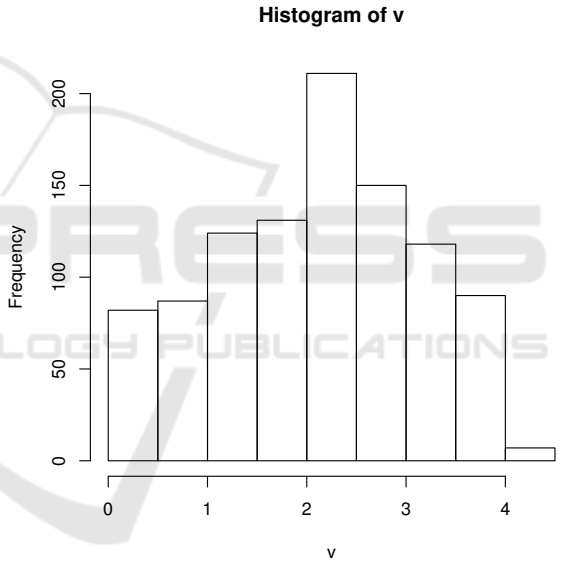

Figure 4: The distribution of the initial *Dev* among peers.

Here we report our main trial performed with a sample of $n = 1000$ students, In Fig. 4 the sample distribution is shown in the (K, J) space while in Fig. 2 the teacher grading distribution shows the gaussian shape of the sample. The experimental plan consists of several runs of the learning algorithms until a final condition is met. The final condition is that difference between two consecutive variations of the network is below a small pre-set quantity. So, the experimental plan is composed by the following steps:

1. A sample of $n = 1000$ students is generated with a uniform distribution in peer assessments. The c rand() function was used;
2. The teacher selects n (in our case n=3) students to

Table 1: The ranked list of the generated sample: on top the highest *Dev* values.

St-ID	K	J	Dev	St
997	4	0,478	4,24	NO_CORE
998	7	0,21	4,24	NO_CORE
999	7	0,38	4,24	NO_CORE
...	...	...	...	
723	6,7	0,42	2,86	NO_CORE
724	4,7	0,26	2,86	NO_CORE
725	4,3	0,38	2,86	NO_CORE

Table 2: A comparison between the grades distributions.

	μ	σ
Teacher	5,51	2,8
Students	6,43	1,66

Table 3: A comparison between the grades distributions.

	μ	σ
Teacher	5,51	2,8
Students	6,02	1,46

grade from the ranked list;

3. All the SMs are updated according to the algorithms shown in SEct. 3;
4. The K-NN algorithm is launched;
5. The new statistical general parameter are computed.

The steps 2-4 are launched several times, until the final condition is met.

After 4 K-NN runs and 8 teacher grades, we obtained the results shown in Tab. 3. The teacher gave a 5.51 mean grade, with $\sigma = 2,8$ while the peers a more generous 6.43 with $\sigma = 1,66$. The initialization of the system started from a mean $\mu = 6.43$, then developed to 6.02 after the k-NN steps. One key point, in our opinion, is in the standard deviation of the assessments, which is diminishing with the k-NN step. This seems encouraging, as it suggests that the framework can improve on the pure peer-evaluation, and also produce more stable assessment distributions.

5 CONCLUSIONS AND FUTURE WORK

In this article we presented a peer assessment system based on a modified version of the K-NN algorithm. We have included the random generation of SMs distribution and some changes to the formulas at the base of the student model's evolution, i.e., learning. The experimental results are encouraging: the system cold help teachers to manage big numbers of students. As future developments we plan to expand the possibility of simulating students with other statistical distributions and then calibrating the learning mechanism. Another perspective regarding future developments concerns the possibility of making the student community evolve autonomously without the teacher's intervention, but based only on social network analysis.

REFERENCES

Anson, R. and Goodman, J. A. (2014). A peer assessment system to improve student team experiences. *Journal of Education for Business*, 89(1):27–34.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Cross, K. P. (1981). *Adults as Learners*. Jossey-Bass, San Francisco.

De Marsico, M., Sciarrone, F., Sterbini, A., and Temperini, M. (2017a). The impact of self- and peer-grading on student learning. *EURASIA Journal of Mathematics, Science and Technology Education*, 13(4):1085–1106.

De Marsico, M., Sterbini, A., Sciarrone, F., and Temperini, M. (2017b). Modeling a peer assessment framework by means of a lazy learning approach. In Huang, T.-C., Lau, R., Huang, Y.-M., Spaniol, M., and Yuen, C.-H., editors, *Emerging Technologies for Education*, pages 336–345, Cham. Springer International Publishing.

Falchikov, N. and Goldfinch, J. (2000). Student peer assessment in higher education: A meta-analysis comparing peer and teacher marks. *Review of Educational Research*, 70(3):287–322.

Kane, L. S. and Lawler, E. E. (1978). Methods of peer assessment. *Psych. Bull*, 85:555–586.

Limongelli, C., Lombardi, M., Marani, A., and Sciarrone, F. (2013). A teacher model to speed up the process of building courses. In *Human-Computer Interaction. Applications and Services - 15th International Conference, HCI International 2013, Las Vegas, NV, USA, July 21-26, 2013, Proceedings, Part II*, pages 434–443.

Limongelli, C., Sciarrone, F., and Temperini, M. (2015). A social network-based teacher model to support course construction. *Computers in Human Behavior*. Article in Press.

Limongelli, C., Sciarrone, F., and Vaste, G. (2008). Ls-plan: An effective combination of dynamic courseware generation and learning styles in web-based education. In *Proc. AH'08: Adaptive Hypermedia and Adaptive Web-based Systems*, volume 5149 LNCS, pages 133–142.

Mitchell, T. M. (1997). *Machine Learning*. David McKay, New York, NY, USA, 1 edition.

Piaget, J. (1971). *Science of education and the psychology of the child*. Longman, London.

Sadler, P. and Good, E. (2006). The impact of self- and peer-grading on student learning. *Educational assessment*, 11(1):1–31.

Sterbini, A. and Temperini, M. (2012). Dealing with open-answer questions in a peer-assessment environment. In *Proc. ICWL 2012. LNCS, vol. 7558*, page 240248.

Vygotsky, L. S. (1962). *Thought and Language*. MA: MIT Press, Cambridge.