Evaluating a programming problem recommendation model - a classroom personalization experiment

André Prisco, Rafael dos Santos, Álvaro Nolibos and Silvia Botelho
Center of Computational Sciences (C3)
Federal University of Rio Grande (FURG)
Neilor Tonin
Integrated Regional University
Erechim Campus
Jean Bez
Federal University of Rio Grande do Sul (UFRGS)

***Abstract*—In this full paper, research to practice, we present a classroom experience, in which we apply a teaching personalization model in an introductory computer science class. Students in this discipline are freshmen at the university and have different backgrounds related to solving programming problems. The traditional approach is standardized, tending to not serve each student in the best way and that is why we have adopted this group as a case study. We use the ELO-based model to recommend specific learning objects for each student, in order to match the student's ability with the difficulty of the problem. The learning objects correspond to programming problems in an online platform for automatic submission and evaluation. The experiment was divided into three stages. In the first, the student was able to freely choose problems from the platform repository. In the second stage, problems were randomly recommended (as a control). In the third stage, the recommendation was made using the model adopted. Students were encouraged to give feedback on their experience described in a free text and in the labeling of hashtags about the learning object. In addition, the rates of success, error, withdrawal and the frequency of access to the online platform were also collected. We observed that the students had a higher engagement (in terms of a higher frequency of use, a higher hit rate, and the production of positive feedbacks) at the stage when the recommendation matched the proposed model.**

***Keywords*—IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

Personalization is a current educational objective. In recent years, information technology has provided subsidies so that computational tools can infer different mediations for different students according to their profile. This profile can also be collected automatically through tools currently available to students.

In this work, we present an educational experiment with the objective of validating a model for recommending programming activities. The experiment consists of using a computer system (developed by us) with a chat interface integrated to an online judge environment, with a repository of programming problems. The system sends the student personalized activities to his profile. The student analyzes the problem and sends his answer (a source code) to the online judge. The system monitors its activities to solve the problem and uses this information (from this and other users) to update the profiles and make new recommendations, so that the use of the system feeds the metadata necessary for its operation. The collected data is used to obtain information about the students' performance and engagement.

We apply a statistical model based on the ELO rating [1], [2], which is based on the history of user interactions and the result of such interactions. In the proposed model, the profile emerges from this data, requiring a minimum of initial metadata or filled in by specialists. As the number of interactions increases, the accuracy of the model in defining the profiles increases. The model is especially useful in massive educational environments, which require less human intervention and can provide more data. The model can be applied in several other educational environments, being a technological contribution to the area.

Unlike our previous work, in which we analyzed the online judge's LOG of interactions to test the model, in this work we present an intervention experiment, recommending problems for computer engineering students, where we were able to evaluate their performance. Freshmen of computing course at our university come with different educational backgrounds, with technical course students, self-taught students and students without any programming knowledge. This heterogeneity is an interesting environment for testing personalization. The experiment methodology, described in the section III, can also be seen as a classroom experience that can be replicated by applying academic innovations.

In the next sections we present an overview of the proposed model, the methodology of the experiment and the results obtained. Finally, we present some conclusions and future work.

II. Performance Expectancy as a tool for recommendation of learning objects

In previous works, we used data from Online Judges systems to create a model to be used in a recommendation system. Online Judges are systems designed to support events such as the programming marathon, Official ACM events [1]. At these events, computer students form teams to compete against each other solving programming problems. The competition is

[1] Association for Computing Machinery

well known and students from all over the world participate in it.

The system used in this work is a virtual environment that has a repository of programming problems, in addition to discussion forums, classification of learning objects by level of difficulty and category and diagnosis of monitoring their evolution.

This tool is available for any student to practice programming problems. Thus, teachers and students began to incorporate the use of the system not only for competitions, but as a didactic tool. The database used in the work has more than 1500 problems and 80,000 users. In this tool, the student can submit his/her solution to a certain problem (learning object) and receive a feedback about it.

To create our model ([1], [3]), we used the ELO rating as the basis, originally created for games [4].

Let R ={0,1} be the set of the results of a match. 1 for victory, 0 for defeat. Given a match between the players i and j, with ELO_i and ELO_j, respectively. The expectation of i to win j ($R_{i,j} = 1$) is given by the equation 1. The ELO Update is given by the equation 2, with K being a constant that adjusts the importance of a match (the higher the K, the greater the potential ELO variation as a result of the match). The logistical parameter (in the example, 400), indicates how discriminatory the difference between ELOs is. In our case, a difference of 400 ELO points indicates that the highest ELO would have a 90% chance of winning. We chose these values based on how the related works configure the initial values in similar environments [2], [5].

The central idea of the update is the performance expectation and the break of expectation. If a player faces a weaker opponent and wins, the variation should be small. If the player faces a stronger opponent and wins, there is a drop in expectation and the ELO variation is greater. This makes winning against stronger opponents more rewarding than weaker opponents.

$$P(R_{ij} = 1) = \frac{1}{1 + 10^{\frac{ELO_j - ELO_i}{400}}} \tag{1}$$

$$ELO_i = ELO_i + K(R_{ij} - P(R_{ij} = 1)) \tag{2}$$

We do not use ELO as a competitive tool, but to model the student-problem relationship. We consider each submission as a game, a *interaction* between the student and the problem.

As discussed in the genetic epistemology [6], [7], the interaction allows modifying and manipulating the object to be learned, in the same way that it modifies the individual and the adaptation of the individual to the object / problem is learning.

We seek to model this concept based on the variation of students' ELOs and problems. Our base has stored the ELO of each problem and student. A correct submission indicates that the student "won the game" while an incorrect submission indicates that the "problem has won". After the assessment, the ELOs are updated, obtaining, in the end, values of relative skill level for students and problems.

Based on these relative skills obtained, the model proposes to identify areas where there is a great potential for equilibration/reequilibration of the student's mental structures [8], enhancing learning. This potential is associated with three possible scenarios. When the potential is small, the interaction is unlikely to unbalance the student, contributing little to his learning. When the potential is too great, although imbalance is likely, the chance of reequilibration (and consequently learning) is small. For a potential for moderate imbalance, we understand that there are good chances of the interaction promoting the subject's reequilibration and therefore learning. In our model, we relate the zone to the Δ_{ELO}, which is the difference between the ELO of the problem and the ELO of the student [1]. Figure 1 illustrates how learning objects can be ordered by Δ_{ELO} and chosen within a recommendation zone, personalized for the student and his or her stage of development in programming.

Analyzing the database with the history of student interactions between 2012 and 2016 (4.8 million submissions), we obtained good results (presented in [1], [3]). However, this experiment did not come from any intervention, nor did it recommend problems to any student, we just observed the behavior patterns and the free choices of the participants. That is why we developed the experiment presented in this work, which is a proposal in which we actually recommend programming problems to students.

Some related works used ELO in the classroom. However, they worked with different methodologies, as proposed in [9] or dealt with subjects that involve more memorization than reasoning (which does not apply directly to our educational model), as presented in [10]. In this experiment, we observe not only the pattern of successes and errors, but the analysis of the student's motivation, which involves verifying how long it takes to see the messages, if he goes in the virtual learning environment to analyze the problem and if he does any attempted solution. Students can at any time withdraw from participating and interacting with the referral system. Therefore, going to the end of the experiment is also a metric to be analyzed.

The chat interface as a platform for the experiment is also a way to enrich the analysis of the model. Such an interface helps to collect the necessary data for the motivation metrics and gives us support to interact with the student regardless of their presence in the learning environment.

III. Methodology

In this section we present the methodology in the style of a narrative of conducting the experiment. In the following section, we present the results obtained. The experiment was applied to a group of computer engineering students (freshmen), in the discipline of algorithms. 42 students participated in the process, which took place from September 2019 until October 2019.

A. Calibration of the recommendation system

The initial calibration consists of assigning an ELO value for database problems and an initial ELO value for students. This is usually a bottleneck for recommendation systems that

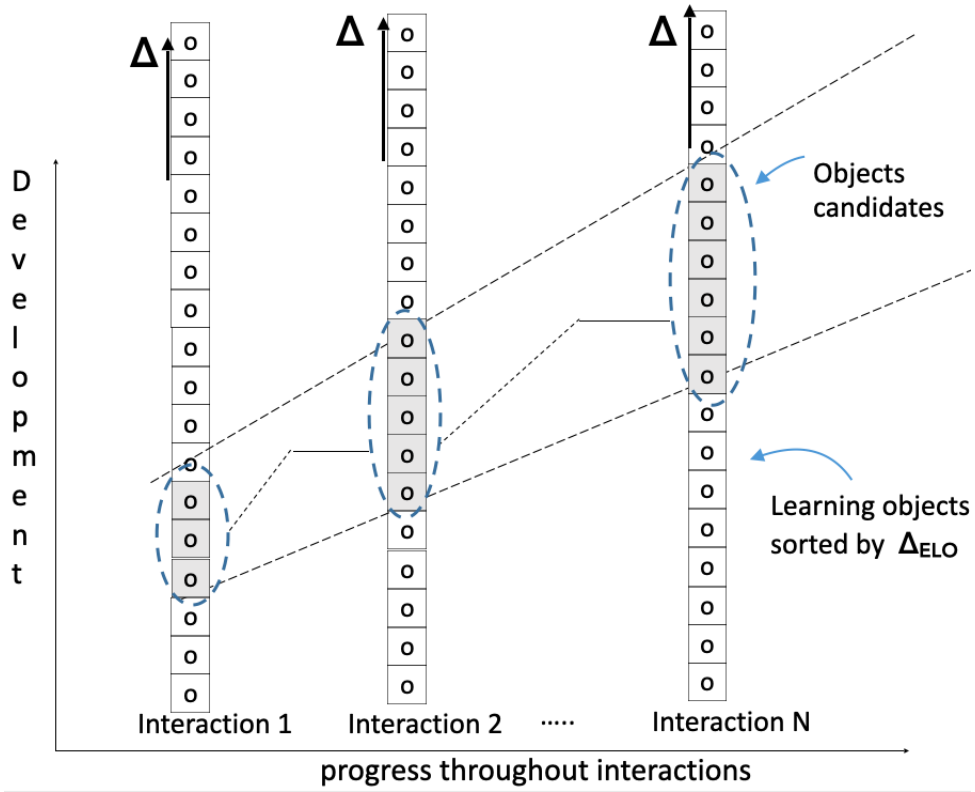

Fig. 1. Recommendation zone - Objects are ordered by Δ_{ELO} at each student interaction. The dotted lines indicate a "window" in which the objects have an appropriate Δ_{ELO} to be recommended. In the experiment, we recommend objects within this range.

are based on behavior analysis, which some authors call "cold start" [11], [12].

We use the online judge's LOG of submissions to calculate the ELO of each problem, considering each past submission as a game, as previously mentioned [1], [3]. At the end of the process, we had an appropriate ELO value (with values between 600 and 2100). Students selected for the experiment and who had already used the environment also had their ELOs calculated from their past submissions. Novice students received an average ELO 1100.

Once calibration is done, the experiment begins.

B. Experiment

The students were instructed about the experiment, that during the process they would receive personalized indications of programming problems, part of a test of a recommendation system. Students enrolled in the online judge environment in the classroom and enrolled in the recommendation system. After this registration step, students were instructed to access the online judge's problem repository and choose 3 problems they felt capable of solving. To support their choice, students could freely read the problem statements and see their classification in the environment. The classification involves the level of difficulty (given by the author of the problem), the category, the success rate. They could also ask for help from colleagues. The only requirement was to choose a problem that he had not previously solved. Each student informed via chat which problems they chose.

From this stage on, the recommendation system works as shown in Figure 2. Students start by interacting with the chosen problems by submitting their solutions to the environment. On our server, we have a crawler that monitors the submission of these students, recording the result of the interaction (successful or unsuccessful) and the moment it occurred. Depending on the result of each submission, we update the student's ELO and the problem in question on our basis.

At the end of a week, or as soon as he gets the first positive result, the student receives a message via chat to fill in a simple feedback about his experience with the problem. We call *submission* each of the attempts recorded by the environment and *interaction* the student's relationship with the problem, containing reading, submissions and feedback. We planned that the feedback should be quite simple, in order to keep the student motivated to fill it out. At the same time, it should be able to collect the student's experience. The Protus [13] system uses the *social tagging* technique. Students are encouraged to, after interacting with a learning object, create a label on that object. Similar to *hashtags* on social networks, students create short terms that describe their experience with the object. The authors state that the use of tags leads students to synthesize the concepts worked on as well as their feedback. We use this approach because it facilitates data analysis and has a didactic function. Related works point out the accuracy and limitations of social tags [14], [15], [16].

After the feedback stage, the student receives the recommendation for two programming problems via chat. From there, we monitor when he will access the problem and make

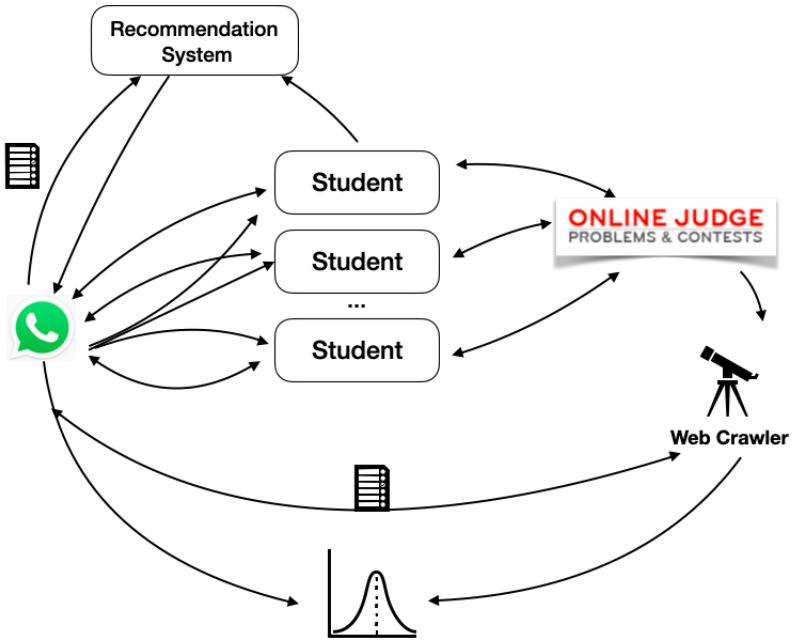

Fig. 2. Experiment - Functioning of the recommendation and data collection system. Students receive recommendation and their behavior in the online judge is monitored.

his submissions. Submissions update the problem and student ELO, which restarts the cycle.

The recommendation criteria vary according to the stage in which we are working. In stage 2, we have a random recommendation. The system chooses any two problems and sends them to the student, regardless of his abilities. In stage 3, we have the recommendation for Δ_{ELO}. We select problems with Δ_{ELO} about of 10 points ($0 < ELO_{object} - ELO_{student} < 10$) and randomly choose one from this subgroup. Students do not know when they are receiving recommendations from stage 2 or 3.

C. Data to be observed

The objective of the experiment is to verify whether a recommendation system based on the Δ_{ELO} model can actually act on the students' engagement and learning process. For this, the following variables are analyzed:

- What is the average time between receiving the recommendation and starting to solve the problems and make attempts?
- What is the average time between a student's attempts, until he gets a correct answer?
- What is the student's performance at each stage (success rate, dropout rate)?
- What is the experience of students when interacting with the programming problem?

With regard to performance, we consider *dropout* when the student does not interact with the problem or makes only attempts with the wrong result, without making new attempts during the recommendation period. *Hit* when the submission was positive and *Error* when the result was negative. When an interaction ends with a positive result, but is preceded by some wrong submissions, we call such submissions *attempts*.

We seek through this methodology to analyze such questions. The results obtained are presented in the following section.

TABLE I. PERFORMANCE RESULTS

Stage	1	2	3
Recommendation	Auto	Random	Δ_{ELO}
Interactions	114	84	92
Attempts	277	28	103
Average number of submissions per interaction	2,43	0,3	1,1
Dropout	49 (43%)	76 (90%)	65 (70%)
Hit	65 (57%)	8 (10%)	27 (30%)

IV. RESULTS

The experiment resulted in 2046 submissions to the online judge environment. These submissions are grouped into 296 problem-student interactions. Students filled out 125 forms, reporting on their experience with the problem.

Table I presents data on the number of interactions and the number of different outcomes for each step.

The first stage, of self-recommendation, resulted in 277 attempts. This number was higher because in the first stage we asked students to choose 3 problems, while in the later stages 2 problems were recommended. However, even with this difference, we identified that the students made more attempts (submissions) to the problems and had a higher success rate than the other stages. These first data suggest that students looked for problems that were new to them but that were within their reach. Reading the statement of the problems and having access to the categorization and level of difficulty, they were able to have some security in their choice.

TABLE II. ANALYSIS OF ACCESS MOMENTS

Stage	1	2	3
Recommendation	Auto	Random	Δ_{ELO}
first attempt	73h	57,5h	41h
hit	11h	21h	7h

Stages 2 and 3 present the problems chosen at random and those actually recommended, respectively. Note that the dropout rate at the random stage is very high (90%). Many of these students ended up giving up on the experiment even before they received an effective recommendation (with repercussions on the increase in dropout at stage 3). At stage 2, students who tried also made few attempts before abandoning problems, averaging 0.3 attempts. When the problem was recommended, the student tried on average more than once (1.1 times). These data indicate greater engagement when the problem is recommended.

Table II shows the time between receiving the recommendation, accessing the problem instructions, and the interaction with the problem.

The student feels curious and takes less time to interact with the problem when it challenges him. When the problem does not adequately challenge the student, or gives the idea of too big a challenge, this student starts to procrastinate his task, taking longer to start interacting. Therefore, the time between the recommendation and the interaction is an indicator of engagement caused by the problem.

Note that at stage 3 students take an average of 40 hours to start interacting. Once the process starts, it takes an average of 7 hours to get a correct answer. This shorter time occurs because the student became involved with the problem in that period, making more than one submission at shorter intervals. In the case of the random stage, the student takes longer to start interacting and even more to arrive at the right answer, which indicates a greater spacing between attempts.

In addition to this data, we ask students to complete a form by inserting hashtags about their experience with the problem and a text field for further details. Filling in these data was not mandatory, but students were encouraged to fill it out. Each student-problem interaction resulted in a form. We received 58, 34 and 30 forms for steps 1, 2 and 3, respectively. On the random stage, we received feedbacks indicating that the recommendation could be wrong for delivering problems that are too difficult or too easy.

One student delivered the same feedback for all of his interactions. We later researched that he had cheated, copying his answers. This occurred even after students were instructed that their performance in the experiment would not be evaluated, but only their participation. Although this case did not hinder the experiment, we consider it a bias some students considered the experiment more as an evaluation than as a study tool.

The Figures 3, 4 and 5 show, respectively, the word clouds resulting from the hashtags of stages 1, 2 and 3. Note the predominance of the word "hard" in stage 2, followed by other terms that indicate a negative experience. Most stages 1 and 3 indicate a positive experience, with no substantial difference.

V. CONCLUSION AND FUTURE WORK

The use of the recommendation system has brought benefits to students in terms of engagement and performance. We emphasize here that a hit on a specific submission is not the best learning criterion. We must observe the trajectory and how the student interacts with the challenge and the feedback that this student adds. Presenting a problem/challenge based teaching process is an effective approach, as indicated by other authors [17], [18]. In this teaching model, the content is implicit in the challenge and it is up to the student to seek it for a specific objective. This pedagogical mediation in this case is to create a context that catalyzes learning.

Despite the effectiveness of the human recommendation (from the student and colleagues), it is not always feasible. In other environments, there is not enough information about the level of difficulty of the challenges, categorization, hit rate, etc. We believe that in these environments with little information available, a recommendation system like the one proposed is necessary.

ELO is effective in contexts that involve reasoning and active learning, as is the case with programming problems. In these cases, the genetic epistemology model is viable even for massive educational environments.

Using a Chat platform for a recommendation system has proved to be an interesting approach for the experiment. We could collect data more easily and integrate a familiar environment for the student into our learning environment than if we had applied the recommendation in a Web site. With chat, we do not need the student to access a tool to make a recommendation, which gave us the freedom to control recommendation moments.

We are currently collecting data in a similar experiment, using the mathematical skills for students starting the calculus course as subject. We intend to compare the results obtained in this experiment with the future results.

REFERENCES

[1] A. Prisco, R. dos Santos, S. Botelho, N. Tonin, and J. Bez, "Using information technology for personalizing the computer science teaching," in *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2017, pp. 1–7.

[2] A. N. Langville and C. D. Meyer, *Who's# 1?: the science of rating and ranking*. Princeton University Press, 2012.

[3] A. Prisco, R. Penna, S. Botelho, N. Tonin, J. Bez, and multidimensional elo model for matching learning objects," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.

[4] A. E. Elo, *The rating of chessplayers, past and present*. Arco Pub., 1978.

[5] K. Wauters, P. Desmet, and W. Van Noortgate, "Monitoring learners' proficiency: weight adaptation in the elo rating system," in *Educational Data Mining 2011*, 2010.

[6] J. Piaget, "Intellectual evolution from adolescence to adulthood," *Human development*, vol. 15, no. 1, pp. 1–12, 1972.

[7] F. Kubli, "Piaget's cognitive psychology and its consequences for the teaching of science," *European Journal of Science Education*, vol. 1, no. 1, pp. 5–20, 1979. [Online]. Available: https://doi.org/10.1080/0140528790010103

[8] N. J. Salkind, *An introduction to theories of human development*. Sage Publications, 2004.

Fig. 3. Wordcloud of hash tags - stage 1 - The words were translated from portuguese to facilitate understanding

Fig. 4. Wordcloud of hash tags - stage 2 - The words were translated from portuguese to facilitate understanding

Fig. 5. Wordcloud of hash tags - stage 3 - The words were translated from portuguese to facilitate understanding

[9] K. Mangaroska, B. Vesin, and M. Giannakos, "Elo-rating method: Towards adaptive assessment in e-learning," in *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, vol. 2161. IEEE, 2019, pp. 380–382.

[10] R. Pelánek, "Applications of the elo rating system in adaptive educational systems," *Computers Education*, vol. 98, pp. 169 – 179, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S036013151630080X

[11] H. Drachsler, H. Hummel, and R. Koper, "Personal recommender systems for learners in lifelong learning: requirements, techniques and model," 2007.

[12] A. Klašnja-Milićević, M. Ivanović, and A. Nanopoulos, "Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 571–604, 2015.

[13] A. Klašnja-Milićević, B. Vesin, and M. Ivanović, "Social tagging strategy for enhancing e-learning experience," *Computers & Education*, vol. 118, pp. 166–181, 2018.

[14] S. Bateman, C. Brooks, G. Mccalla, and P. Brusilovsky, "Applying collaborative tagging to e-learning," in *Proceedings of the 16th international world wide web conference (WWW2007)*, 2007.

[15] D. Verpoorten, M. Chatti, W. Westera, and M. Specht, "Personal learning environment on a procrustean bed–using plem in a secondary-school lesson," in *Proceedings of the London international conference on education (LICE-2010)*. LICE, 2010, pp. 197–203.

[16] A. Klanja-Milicevic, B. Vesin, M. Ivanovic, and Z. Budimac, "Personalisation of programming tutoring system using tag-based recommender systems," in *2012 IEEE 12th International Conference on Advanced Learning Technologies*. IEEE, 2012, pp. 666–667.

[17] G. Sharma, "Effectiveness of problem-solving teaching technique on the evolvement of higher level learning outcomes." *Psycho-lingua*, 2000.

[18] S. M. Souza and R. A. Bittencourt, "Motivation and engagement with pbl in an introductory programming course," in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–9.