Evaluation of a hybrid AI-human recommender for CS1 instructors in a real educational scenario

Filipe Dwan Pereira[1,5], Elaine Oliveira[3], Luiz Rodrigues[4], Luciano Cabral[5,6], David Oliveira[3], Leandro Carvalho[3], Dragan Gasevic[7], Alexandra Cristea[2], Diego Dermeval[4], and Rafael Ferreira Mello[5,7]

[1] Federal University of Roraima filipe.dwan@ufrr.br
[2] Durham University
[3] Federal University of Amazonas
[4] Federal University of Alagoas
[5] CESAR School
[6] Federal Institute of Pernambuco
[7] Monash University

Abstract. Automatic code graders, also called Programming Online Judges (OJ), can support students and instructors in introduction to programming courses (CS1). Using OJs in CS1, instructors select problems to compose assignment lists, whereas students submit their code solutions and receive instantaneous feedback. Whilst this process reduces the instructors' workload in evaluating students' code, selecting problems to compose assignments is arduous. Recently, recommender systems have been proposed by the literature to support OJ users. Nonetheless, there is a lack of recommenders fitted for CS1 courses and the ones found in the literature have not been assessed by the target users in a real educational scenario. It is worth noting that hybrid human/AI systems are claimed to potentially surpass isolated human or AI. In this study, we adapted and evaluated a state-of-the-art hybrid human/AI recommender to support CS1 instructors in selecting problems to compose variations of CS1 assignments. We compared data-driven measures (e.g., time students take to solve problems, number of logical lines of code, and hit rate) extracted from student logs whilst solving programming problems from assignments created by instructors versus when solving assignments in collaboration with an adaptation of cutting-edge hybrid/AI method. As a result, employing a data analysis comparing experimental and control conditions using multi-level regressions, we observed that the recommender provided problems compatible with human-selected in all data-driven measures tested.

Keywords: Hybrid systems evaluation · recommender system · introductory programming.

1 Introduction

Programming Online Judge (OJ) is an auto-grader system capable of providing instantaneous feedback about the accuracy of a code solution [30, 11, 21].

Typically, these systems offer programming problems of different topics related to computer science subjects [30, 34, 11]. Computer Science 1 (CS1) instructors are increasingly adopting OJ to enhance the learning experience in the offerings of this introductory course [15, 22, 30, 33]. For the instructor, it reduces the workload due to the automatic evaluation provided by the OJ [30, 20]. For the student, it allows them to readily correct their code errors (due to the instantaneous feedback) and receive a fair assessment ([16] demonstrate that OJ assessment is fairer than the human one). The OJ feedback can highlight the strengths and weaknesses of the learners' solutions, allowing them to improve their skills and knowledge.

Nonetheless, there is much room for improvement in OJ systems to support CS1 classes [30, 34, 20]. To illustrate, due to the wide variety of problems provided in these systems, it is hard for the instructors to select appropriate problems to compose assignments. As such, it could take much time for the instructors to create the assignments. In such situations (overload of options – problems), the combination of OJ and recommendation systems are suggested in the literature [32, 34, 20, 30].

Despite calls from literature [30, 28, 15, 17, 33] for research about recommending problems in OJs, only a few works have proposed methods for this task. Furthermore, the few methods available are generally suitable for expert students, but not for novices [30, 32, 20, 34]. It is worth noting that proposing methods to assist instructors of CS1 is claimed to be crucial for the advancement of learning to program [17, 27].

In this work we replicated, adapted and validated a cutting-edge method [24] to help instructors select problems in a CS1 course, where the instructors submit a list of problems to a recommender system, which suggests similar problems that the instructors can use, for instance, to avoid plagiarism. The method was validated in a novel data-driven setting, with two conditions: one where students received problems generated by the instructors, and one where students receive problems selected by the instructors using the AI recommendations.

2 Related Work

The methods proposed in the literature to recommend problems in OJ typically investigate only shallow features based on the students' submissions and attempts to perform the recommendation. For instance, Fantozzi and Laura [9] proposed a recommender system for expert users who are training for the International Collegiate Programming Contest (ICPC)[8]. The method employs students' attempts at an OJ and uses an autoencoder neural network to recommend problems. Similarly, Saito and Watanobe [28] proposed a recommender that applies a recurrent neural network to produce students' learning paths based on their attempts. Yera and Martínez [32] also applied students' attempts and feature engineering to create a recommender system for OJs. All these works [9, 28, 32] demonstrated results only in a synthetic scenario based on machine

[8] icpc.global

learning performance metrics (e.g., F1-score). Evaluations with real users were not carried out. Moreover, these methods are best suited for expert learners, particularly those who are training for the ICPC.

Recent works [7, 20] suggest tracking log information (instead of only collecting the number of attempts) when students are solving problems in an Integrated Development Environment (IDE) to extract fine-grained features that depict the effort students take to solve the problems. Carter et al. [7] explain that IDE offers researchers the widest spectrum of student process data, as this is where learners spend a large majority of their time problem-solving. An example is [20], which proposed a recommender system in OJs based on the effort required to solve problems focused on CS1 students. The features employed were based on a data-driven analysis of how students solved problems whilst they were developing their code in an IDE. Despite finding satisfactory results when compared with a baseline, the authors pointed out limitations related to the lack of information about the problems' topics. That is, they observed that it was necessary to detect the topic of the problem and associate this information with the effort required to solve the problem.

In this sense, a recent work [24] extended the work proposed in [20], adding a topic detector and caring out a validation of the recommender with 35 CS1 instructors. The results surpassed the baseline (an adaptation of Yera and Martínez's work [32]) in the measures experimented. The validation was carried out through a questionnaire in which the instructors estimated the coding effort, hit rate and resolution time expected to solve the automatically recommended problems versus questions recommended by their baseline [32]. Nonetheless, despite the importance of validation with instructors and their satisfactory results in a laboratory setting, Fincher et al. [10] explain that it is critical to evaluate educational methods in real scenarios (scarcity of proposed methods evaluated in real scenarios), and with students. Schwartz and Gurung [29] claim that it is imperative to go beyond questionnaires and validate methods available in the literature using data-driven evidence as well.

In this regard, de Oliveira et al. [19] evaluated whether extended feedback from an OJ system would improve the students' motivation using data-driven evidence collected from the students' side. Still, Quille and Bergin [26] evaluated how a performance prediction system associated with an OJ could impact the students' outcomes. Both works performed that analysis in a controlled experiment in a real educational scenario. However, there is a lack of studies evaluating how recommendation systems from a data-driven perspective work in a real educational scenario with CS1 students.

Furthermore, previous works [19, 26] imply learning improvements when the hybrid human/AI methods were associated with instructor mediation. That is, the AI method might recommend options for the instructor who decides how the AI information should be applied to enhance the students' outcome. Moreover, the instructors could also enhance the AI method capabilities. To understand deeply the importance of those methods, the students' perspective must have been observed and analysed [13, 24]. However, to the best of our knowledge,

there is no work that evaluates a recommender system to support CS1 students from the perspective of the students, and with data-driven evidence.

3 Contributions and novelty

The study described in the current paper adapted the state-of-the-art recommendation method proposed in [24] and validated it in a real scenario, with 7 CS1 classes, and 196 students. Our proposal performs a detailed analysis of the recommender system under different aspects: i) employing a data-driven evaluation of the method by collecting fine-grained data whilst students solved the recommended problems in an IDE embedded in the OJ; ii) comparing the problems recommended by the method with problems recommended by humans; and iii) adapting a hybrid approach for the recommendation, in which the instructor validates and rectifies the recommendations provided by the AI and also enhance the AI capabilities. As such, this study is an incremental work that represents a step towards validating and employing a cutting-edge method available in the literature, by, for the first time, to the best of our knowledge, replicating their work [24] using their open dataset and validating their method in a real-life scenario over a data-driven perspective from the students' perspective.

Thus, the main contribution of the paper is the practical application of an educational method available in the literature [24] in a real-world educational scenario and testing and evaluating their effectiveness with data-driven approaches. This allows for refining and improving the method, and ensuring that it is effective in diverse settings and for a wide range of users. Moreover, this offers valuable evidence for the future of OJ systems to support CS1 instructors and students.

4 Educational Scenario

Introduction to Programming is offered at the Federal University of Amazonas as part of several undergraduate courses. The objective of the discipline is to help students learn to solve algorithmic problems and to offer the ability to elaborate, verify and implement algorithms in a high-level programming language.

Since 2020, the research developed in this area is being supported by the SUPER Project, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon.

These CS1 courses cover 7 topics in the CS1 assignments. As the topics are cumulative, content from earlier topics appears in later topics. Each assignment contains an average of 12 problems related to the following topics: (i) sequential structure; (ii) simple conditionals (if-then-else); (iii) nested conditionals (if-then-else nested); (iv) while loops; (v) vectors and strings; (vi) for loops; and (vii) matrices. Exams generally are composed of 2 questions. To avoid plagiarism in the assignments, the instructors create variations of assignments each semester, using distinct questions. The methodology (e.g., pedagogical material and notes) used in CS1 classes is the same, regardless of the instructor who teaches the

course. The assignments are carried out using the tool CodeBench[9], an online judge system used by instructors from different universities that performs an automatic evaluation of the students' code solutions. After instructors post the assignments to CodeBench, students solve problems in an IDE embedded into CodeBench. Python is used as the target programming language of the course.

5 Recommendation Method

For the recommendation, we adapted the hypothesis presented in [24]: *if students have solved a given problem (Target Problem (TP)) and there are problems similar to TP in terms of topic and effort required, thus these problems can be used as potential Recommended Problems (RPs) to replace TP in new assignments.* To validate that hypothesis and replicate the work in [24], we used the CodeBench open dataset, which is described in the next section.

5.1 Dataset

CodeBench provides a freely anonymised dataset[10] with fine-grained information on programming students' behaviours. Whilst students are solving problems in an IDE embedded in the CodeBench system, the students' logs are collected. In short, the dataset contains each character typed by the students and the timestamps. It is also possible to know such information as when the students stopped typing, when they pasted content in the IDE, and when they clicked outside the IDE.

Following the proposal given in [24], we extracted 22 features from the students' logs to feed the recommender system. The features include the hit rate students take to get problems accepted, the average lines of code per problem, the average time students take to solve problems, the average cyclomatic complexity of the solutions per problem, etc. The entire list of features is available in [24].

Moreover, we used the problem descriptions to detect the problem topics based on the CS1 syllabus, as suggested in [24, 23]. To feed the recommender, we used the data from 2016 to 2019 (before the pandemic[11]), as can be seen in Table 1. Furthermore, the number of different problems from CodeBench we used in this work is 1,026, as shown in Table 2. The table shows the problems segregated by the CS1 topics. These are the problems the recommender system can choose from.

[9] codebench.icomp.ufam.edu.br

[10] codebench.icomp.ufam.edu.br/dataset/

[11] During the pandemic, the course stopped for a while and after 1 year, it was reoffered remotely, instead of face to face. That's why we do not use the data during the pandemic, since it is in different educational conditions.

Table 1: CodeBench's data set segregated by term. Assig. Probs means number of assignment problems, whereas Ex. Probs. means Exam problems.

Term	Students	Assig. Probs	Ex. Probs.	Code	Attempts
2016-1	471	681	124	30140	154163
2016-2	172	447	110	9501	38933
2017-1	463	1278	163	38304	119370
2017-2	177	556	103	10942	27613
2018-1	465	1550	182	47969	148775
2018-2	180	893	107	13458	46765
2019-1	489	1559	176	38417	140537
2019-2	297	1116	138	28348	36632
Total	2714	8080	1103	217079	712788

Table 2: Description of CodeBench Topics.

Topic	Description	N
sequential structure	arithmetic operations and use of variables	157
if-then-else	conditional structure	136
if-then-else (nested)	nested conditionals structure	161
while-loop	loops using the structure while	114
for-loop	loops using the structure for	117
vectors and strings	unidimensional vectors and string operations	207
matrices	bidimensional matrices operation	134
Total		1,026

5.2 Recommendation steps

Figure 1 shows how the recommendation is performed by this recommender method. To summarise, the instructor defines the TP that will be used as a reference problem to perform the recommendation. In practice, the TP is a given problem previously used by the instructor in an assignment provided by classes from previous semesters. The recommender first detects the topic of the TP, based on the CS1 syllabus (see Table 2). The method then recommends an RP that is associated with the same topic as the TP and that requires a similar effort to be solved. For example, if the TP is associated with the conditional structure (if-then-else), then the recommender suggests a conditional structure problem as well that requires a similar effort to be solved.

To detect the topic of problems, we use the pipeline proposed in [23], which combines Natural Language Processing (NLP) techniques with shallow and deep machine learning models (a similar NLP pipeline was effectively used in other cutting-edge works [3, 4]). As a result, the best model (which was a Convolutional Neural Network) reached 90% F1-score. In this case, the need to use a predictive model is that most questions from OJs are not annotated with the CS1 topics.

6 Experimental Study

The focus of this paper is to evaluate the method presented in section 5 in a real scenario, using data-driven evidence from the students' perspective, instead of applying a questionnaire for instructors. This evaluation was divided into two

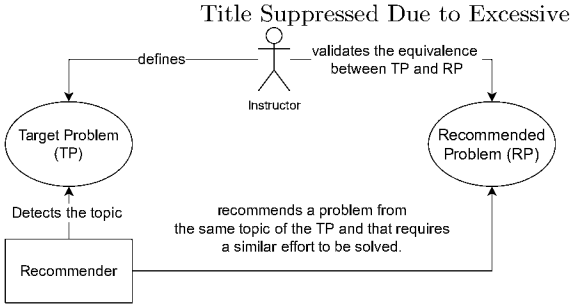

Fig. 1: How the recommendation is performed.

steps: i) the instructors' selection of the problems in collaboration with the AI recommender system (detailed in section 6.2); and ii) the data-driven evaluation of the students' logs when solving the recommended problems, versus when solving problems selected manually by the instructors (detailed in section 6.3).

The data utilised for this data-driven evaluation was collected during the first term of 2021. The course methodology aligns with the pattern outlined in section 4. It is important to note that we have two distinct sources of log data: i) to support the functioning of the recommender system, which was collected between 2016 and 2019 (Section 5.1); ii) to conduct the data-driven evaluation from the students' perspective, which was gathered specifically during the first term of 2021.

6.1 Conditions (Controlled vs Experimental)

The instructors of the CS1 courses that participated in this experiment created seven assignments for the course, one assignment for each topic (see the seven topics in section 4). For the controlled condition, we used the assignments composed by these instructors. For the experimental condition, we used the problems recommended by the method specified in section 5, as illustrated in Figure 1. The assignment lists used in the control condition contained 12 problems. For the sake of simplicity, we adopted the first six problems in these lists to be used as TPs. In this way, the recommender selected the nearest neighbour of each TP to compose the first six problems in the assignment lists in the experimental condition.

More formally, we were interested to assess whether the variation assignments (let us call them List B) were equivalent to the Master List in terms of the effort required to solve the lists and the topic of the problems. Let us call *Master Lists*, the assignments created by CS1 instructors. Our intention, thus, was to assess whether a List B, which is a variation of a Master List, in which the problems from both (Master List and List B) must be equivalent in terms of the topic associated and the effort required to solve the problems. To illustrate, if the first six problems from the Master List are $L_{master} = \{TP_1, TP_2, TP_3, TP_4, TP_5, TP_6\}$ then List B must contain the problems $L_B = \{RP'_1, RP'_2, RP'_3, RP'_4, RP'_5, RP'_6\}$, where each pair of questions (TP_i, RP'_i), i varying from 1 to 6, must require a similar effort and topic to be solved by the students.

6.2 Hybrid Human/AI Recommendation

Attending a call from the literature [13, 8, 1], we adapted the method from [24], to employ the concept of hybrid intelligence, where we combine instructor knowledge with our AI method, to collaboratively create assignments in real classes of CS1.In total, 7 instructors participated in this study. The instructors were responsible for the courses of the control and experimental condition. Following the recommendation steps presented in section 5, the instructors evaluated the problems provided by the recommendation method according to the following criteria: i) if the topics of both problems (TP vs RP) were equivalent; ii) if the estimated time for the resolution of both (TP vs RP) were equivalent; iii) if the estimated coding effort for the resolution of both (TP vs RP) were equivalent; iv) if the estimated hit rate for the resolution of both (TP vs RP) were equivalent; and v) whether the instructors thought the problems (TP vs RP) were interchangeable. These criteria were defined in line with the work of [24]. The criteria under ii) to v) were employed to estimate the effort required to solve the problems and their equivalence, as suggested in [24]. The last criterion (v) was central to the instructor's decision to keep the problem on the assignment list. In total, 10 problems were replaced by the instructors, 5 of them because the pair of problems were from different topics, and 5 because of incompatibility in terms of effort required to solve the problems (one of the items from i) to iv)).

In case of the instructor not accepting the AI method recommendation, a justification is given by the instructor, explaining the reason why the RP was not kept on the assignment list. The justification is given based on the above five criteria (i-v). The justification is then used to adjust the recommender, i.e., to improve the AI. For example, a source of error for the recommender is when the topic detector misclassifies a given problem (this occurs 10% of the time since the model is 90% accurate). When this happens, the instructor can adjust the AI, by relabelling the problems with the correct topic , so that the next time, this RP will be associated with the correct topic. Notice that after the instructor not accepting the RP, a new recommendation is performed.

Regarding criteria i-iv, the system recommends the i-th closest neighbour of the TP, starting the variable i from 2 (that is, from the second nearest neighbour[12]) and increasing this value if the substitution was not accepted by the instructor (for more information about the recommendation process itself, please check [24]). In case of a not accepted recommendation here, the instructor does not rectify the AI-method recommendation. We just use the following nearest neighbour of the TP as the RP. The proposal of instructor adjustments of the AI, in this case, will be made in future work.

As such, human/AI collaboration is performed in three sequential steps: i) the instructor provides as input an assignment (we call it, as said, the Master List - controlled condition), ii) the AI method returns one or several recommended problem(s) for each problem of that Master List to be used in the assignment variation(s) (List B - experimental condition), iii) the instructor validates each

[12] Notice that the first nearest neighbour of a given TP is itself and that is why we start i from the number 2.

recommendation and decides whether they can be used in the assignment/exam variation(s), and iv) AI is adjusted in case of RP replacement (for topic incompatibility).

6.3 Evaluation with students

For the method evaluation, we collected fine-grained data from the students (during the first term of 2021) whilst they were solving the assignment problems of the controlled (Master Lists) and experimental conditions (Lists B). An example of the fine granularity data collected is given in Figure 2, when a student was writing a print command.

The total of problems in all assignments was 42 (6 problems x 7 topics). Fine-grained data (see an example in Figure 2) was collected from students, whilst they were writing their solutions in the CodeBench IDE. A total of 196 students participated in the study, 130 in the control group and 66 in the experimental group. All the data was anonymised and consent for data collection analysis and publication was given by all students who participated in this experiment.

```
27/5/2016@8:34:42:871:focus
27/5/2016@8:34:43:475:change:{"from":{"line":0,"ch":0},"to":{"line":0,"ch":0},"text":["p"],"removed":[""],"origin":"+input"}
27/5/2016@8:34:43:615:change:{"from":{"line":0,"ch":1},"to":{"line":0,"ch":1},"text":["r"],"removed":[""],"origin":"+input"}
27/5/2016@8:34:43:677:change:{"from":{"line":0,"ch":2},"to":{"line":0,"ch":2},"text":["i"],"removed":[""],"origin":"+input"}
27/5/2016@8:34:43:811:change:{"from":{"line":0,"ch":3},"to":{"line":0,"ch":3},"text":["n"],"removed":[""],"origin":"+input"}
27/5/2016@8:34:43:884:change:{"from":{"line":0,"ch":4},"to":{"line":0,"ch":4},"text":["t"],"removed":[""],"origin":"+input"}
27/5/2016@8:34:43:884:change:{"from":{"line":0,"ch":5},"to":{"line":0,"ch":5},"text":["()"],"removed":[""],"origin":"+input"}
```

Fig. 2: Logs collected when the learner was writing a *print* command.

From such data, we extracted the following measures to evaluate whether the TPs and RPs from each condition was equivalent:

- *Resolution Time*: time the student took to solve the question, not counting downtime (more than 2 minutes).
- *codeEffort*: whether the expected coding effort (number of lines of code, number of control structures used) to solve both TP and RP problems are equivalent.
- *hitRate*: if the hit rate expected to solve both TP and RP problems are equivalent. The hit rate is the proportion of code solutions accepted (correct) divided by problems attempted, for a given problem.

6.4 Data Analysis

We compared experimental and control conditions using multi-level regressions [14] because such regression models are aligned to our numeric measures as well as handle dependent data (i.e., multiple answers from the same students). Standard t-tests, for instance, do not handle more than two repeated measures, whilst ANOVAs suffer with unbalanced datasets such as ours [6] and standard regressions assume data points are independent [12]. Differently, multi-level regression models handle dependent data because, loosely speaking, they create a number of standard regression models (e.g., one for each level), and then group

them together. In our case, for instance, the grouping factor was *users*. The algorithm took group differences (e.g., from one person to another) into account by, oftentimes, having one intercept per group. These are known as the *random coefficients* [18].

Additionally, multi-level regressions calculate *fixed coefficients*. These can be seen as overall properties of the dataset that do not vary across groups but are calculated considering group (dis)similarities [18]. Thereby, using multi-level regressions is imperative to properly reach the goal of the proposed study given our experimental design. Note that we were interested in understanding the overall effect of the recommendation method compared to human selections. Therefore, we focused on fixed coefficients, which provide this overall perspective while controlling for data dependency. Thus, we ran three multi-level regressions, one with each measure as dependent variable, and users as the grouping factor (i.e., random coefficients) and condition (control vs experimental) as the independent variable (i.e., fixed coefficients) for all.

Importantly, multi-level regressions have assumptions similar to those of standard regression [14]. Residuals normality, for instance, is one of them [12]. However, testing this and similar assumptions is subject to new assumptions and threads to validity as well [6]. Then, we opted to additionally conduct robust analyses. These are considered to address limitations of the standard approaches in the presence of, for instance, outliers, and yield similar results when original assumptions are met [31]. Therefore, we ran standard and robust multi-level analyses – one pair for each measure – using the *lme4* and *robustlmm* R libraries, respectively. Then, we compared the standard's and robust's results to identify possible differences among them aiming to maximise conclusion validity. We did not correct p-values because these tests represent a small number of comparisons planned apriori [5] and use 95% confidence intervals for our comparisons and p-values, which were calculated using the *lmerTest* R package.

7 Results

Table 3 and Figure 3 summarise our results. First, they show the mean and standard deviation values of each condition for all three measures. Next, they show the multi-level regression results, focusing on the fixed coefficient (experimental vs control) (see section 6.4). Specifically, the table reports the coefficient and its standard error, t statistic, and p-value. Finally, the table reports the 95% confidence intervals estimated for each condition.

As Table 3 demonstrates, all p-values were greater than the alpha level (0.05). Accordingly, the CIs of experimental and control groups overlapped for all measures. Hence, we could not reject the null hypothesis that the condition coefficient significantly affected any of the three measures. Similarly, the robust analyses yielded the same conclusions for hit rate (Coef = 0.01; SE = 0.03; t = 0.245; EXP CI: [0.36 - 0.46]; CTR CI: [0.38 - 0.46]), resolution time (Coef = 2.09; SE = 3.35; t = 0.624; EXP CI: [32.4 - 43.64]; CTR CI: [36.0 - 43.6), and code effort (Coef = 19.6; SE = 13.8; t = 1.428; EXP CI: [158 - 202]; CTR CI: [183 - 215]). Thus, these findings suggest that the recommendation method recommended

Table 3: Results summary for all measures. *Condition* shows descriptive statistics as Mean (Standard Deviation) for experimental (EXP) and control (CTR) groups. *Multi-level Regressions* shows the condition coefficient (Coef) and its standard error (SE), t statistic, p-value, and conditions' Confidence Intervals.

	Condition		**Multi-level Regressions**				
Measure	**EXP**	**CTR**	**Coef (SE)**	**t stat**	**p-val**	**CI Exp**	**CI Ctr**
Hit Rate	0.39 (0.22)	0.43 (0.26)	0.02 (0.03)	0.47	0.63	[0.37 - 0.47]	[0.40 - 0.47]
Res Time	40.7 (34.7)	45.9 (43.5)	5.32 (3.95)	1.35	0.18	[34.0 - 46.6]	[41.1 - 50.2]
Code Eff.	191 (131)	212 (149)	21.3 (14.0)	1.52	0.13	[166 - 211]	[194 - 226]

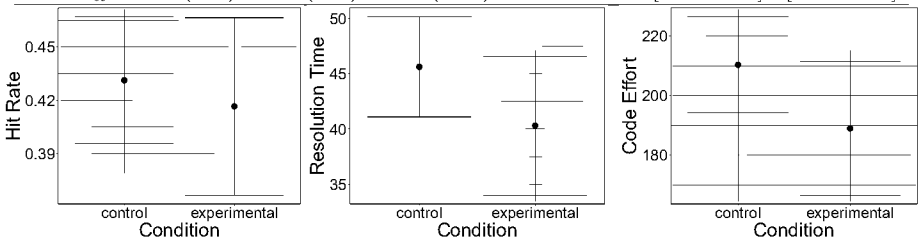

Fig. 3: Outcomes of measures comparing the experimental setting versus the control setting.

problems that required resolution times and coding efforts, as well as led to hit rates, similar to those of problems selected by instructors.

8 Pedagogical Implications

Here we provide an important step towards the employment of a method provided in the literature [24], which is crucial [15, 25] since many AI methods provided in conference and journal papers end up not being used in real educational scenarios. Indeed, our study provides advantages in terms of employing a practical application of an AI method to allow for testing and evaluation of their effectiveness in real-world situations critical for refining and improving this method, and ensuring that they are effective in diverse settings and for a wide range of users. Still, our study enables instructors to tailor and adapt the AI method validated to meet the unique needs of their learners and their specific context. The original work [24] provided general guidelines and principles for the use of their method. However, the implementation of their method requires customisation and adaptation for different contexts, learners, and learning goals. As such, our work allows for such customisation and adaptation, leading to more effective implementation of that educational method.

We strongly believe that the practical application of educational methods available in the literature in real-world educational scenarios is critical for improving educational practice and ensuring that learners receive high-quality and effective education.

Furthermore, a practical implication of our findings is the reduction of instructors' workload in the problem selection task, to compose assignments lists using OJs. That is, save the instructors time and effort by filtering through a vast amount of programming problems and presenting them with only the most relevant ones. This makes room for freeing the instructors' time, enhancing their methodological and pedagogical planning, and reducing the high failure rates that are common in CS1 classes [15, 27].

Moreover, our adaption to employ hybrid human/AI intelligence to the work presented in [24] has opened room for enhancing instructors and AI capabilities for the benefit of both sides. Overall, the combination of AI and human expertise has the potential to enhance the effectiveness and efficiency of education, leading to improved learning outcomes and increased student satisfaction. As such, hybrid human/AI systems are becoming increasingly important in education, and are likely to play a major role in the future of education [13].

Moreover, another possible application for this work is the creation of a pool of similar questions that could be used to identify students with *fragile learning* [27]. Fragile learning refers to a situation where a student may have solved an exercise by chance, without possessing all the necessary skills and concepts needed to solve the problem, or may have found the solution online. A more reliable way to confirm whether the student has actually acquired the skills associated with the exercise is to ask them to solve a similar question, which is similar in terms of topic, estimated time of resolution, and level of effort required. If the student fails to solve a similar problem, it could be an indication of fragile learning, which is important to detect for the evolution of the computer education field [27, 17]. Similarly, asking students to solve similar problems could also be used to identify signs of plagiarism, which is a common issue in CS1 classes [2]. By enabling the creation of tools for detecting fragile learning and preventing plagiarism based on the solving of multiple similar problems, our method offers a valuable contribution to the field.

The current study provided empirical evidence that might help computing education researchers. The method might be also useful to design pre- and post-tests, by creating two or more assignments that require similar effort, and resolution time, and will produce similar hit rates for those with similar knowledge. The challenge to find pairs of equivalent questions to be used in pre- and post-tests of experiments in the area of computer education is well known. Thereby, the method could be intuitively employed for this purpose.

9 Final Remarks

In this work, we assessed a recommendation system used in real-life classes of CS1, to help the instructors in the selection of problems to compose assignment lists. We demonstrated that the humanly validated recommendations (hybrid human/AI approach) are compatible with the humanly selected problems. The method was tested in an environment with real classes, in which students in the control group solved the humanly selected questions, while students in the experimental group solved the selected questions through a collaboration between

the instructor and the AI method. As such, it is believed that the human/AI hybrid method can be employed in online judge systems to reduce the workload of CS1 instructors.

As a limitation, we collected data from a single institution, which brings external threats to the method's validity. Moreover, we obtained the data from our intervention from a single semester, in a teaching cycle of 14 weeks. As such, how our intervention affects the time the instructor takes to select problems, and if it poses more or less challenging problems for students is one aspect that merits more investigation.

Our adaptation of the work from [24] does not provide an alternative in cases where the recommendation is not useful, because of incompatibility between the effort required to solve the TP and RP. In our future work, we plan to enhance our approach by soliciting feedback (or rating) from instructors who choose not to accept a recommendation due to the aforementioned reason (incompatibility between the effort from TP and RP). By asking for a rating of the recommendation, we can gain deeper insight into the extent of its shortcomings and make necessary adjustments to the recommender model accordingly. Additionally, we aspire to conduct a more extensive analysis of CS1 classes to further validate our findings. Indeed, validating research findings with a wider sample is essential for ensuring that the results are reliable and generalisable. By including a larger and more diverse sample, we can improve the representativeness of our findings, and enhance the likelihood that the results will be applicable to other contexts and populations.

Finally, the proposed recommender method could be applied to any programming course. To understand its power of generality in-depth, future research should carry out new experiments in other courses other than CS1.

10 Acknowledgements

This research, carried out within the scope of the Samsung-UFAM Project for Education and Research (SUPER), according to Article 39 of Decree n°10.521/2020, was funded by Samsung Electronics of Amazonia Ltda., under the terms of Federal Law n°8.387/1991 through agreement 001/2020, signed with UFAM and FAEPI, Brazil. This study was financed in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil - CNPq (Process 308513/2020-7) and Fundação de Amparo a Pesquisa do Estado do Amazonas - FAPEAM (Process 01.02.016301.02770/2021-63). This study was financed in part by the Acuity Insights under the Alo Grant program.

References

1. Akata, Z., Balliet, D., De Rijke, M., Dignum, F., Dignum, V., Eiben, G., Fokkens, A., Grossi, D., Hindriks, K., Hoos, H., et al.: A research agenda for hybrid intelligence: augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence. Computer **53**(08), 18–28 (2020)

2. Albluwi, I.: Plagiarism in programming assessments: a systematic review. ACM Transactions on Computing Education (TOCE) **20**(1), 1–28 (2019)
3. Alrajhi, L., Alamri, A., Pereira, F.D., Cristea, A.I.: Urgency analysis of learners' comments: an automated intervention priority model for mooc. In: Intelligent Tutoring Systems: 17th International Conference, ITS 2021, Virtual Event, June 7–11, 2021, Proceedings 17. pp. 148–160. Springer (2021)
4. Alrajhi, L., Alharbi, K., Cristea, A.I., Pereira, F.D.: Extracting the language of the need for urgent intervention in moocs by analysing text posts. In: International Conference on Web-Based Learning. pp. 161–173. Springer (2022)
5. Armstrong, R.A.: When to use the bonferroni correction. Ophthalmic and Physiological Optics **34**(5), 502–508 (2014)
6. Cairns, P.: Doing better statistics in human-computer interaction. Cambridge University Press (2019)
7. Carter, A., Hundhausen, C., Olivares, D.: Leveraging the integrated development environment for learning analytics. In: The Cambridge Handbook of Computing Education Research, chap. 23, pp. 679–706. Cambridge University Press, Cambridge (2019)
8. Dellermann, D., Ebel, P., Söllner, M., Leimeister, J.M.: Hybrid intelligence. Business & Information Systems Engineering **61**(5), 637–643 (2019)
9. Fantozzi, P., Laura, L.: Recommending tasks in online judges using autoencoder neural networks. Olympiads in Informatics **14**, 61–76 (2020)
10. Fincher, S., Tenenberg, J., Dorn, B., H.C., McCartney, R., Murphy, L.: Computing education research today. In: The Cambridge Handbook of Computing Education Research, chap. 2, pp. 40–55. Cambridge University Press, Cambridge (2019)
11. Fonseca, S.C., Pereira, F.D., Oliveira, E.H., Oliveira, D.B., Carvalho, L.S., Cristea, A.I.: Automatic subject-based contextualisation of programming assignment lists. International Educational Data Mining Society (2020)
12. Gelman, A., Hill, J.: Data analysis using regression and multilevel/hierarchical models. Cambridge university press (2006)
13. Holstein, K., Aleven, V., Rummel, N.: A conceptual framework for human–ai hybrid adaptivity in education. In: International Conference on Artificial Intelligence in Education. pp. 240–254. Springer (2020)
14. Hox, J.J., Moerbeek, M., Van de Schoot, R.: Multilevel analysis: Techniques and applications. Routledge (2010)
15. Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S.H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M., Sheard, J., Skupas, B., Spacco, J., Szabo, C., Toll, D.: Educational data mining and learning analytics in programming: Literature review and case studies. ACM. Proceedings of the 2015 ITiCSE on Working Group Reports pp. 41–63 (2015)
16. Kurnia, A., Lim, A., Cheang, B.: Online judge. Computers Education **36**(4), 299–315 (2001). https://doi.org/https://doi.org/10.1016/S0360-1315(01)00018-5
17. Luxton-Reilly, A., Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: a systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. pp. 55–106 (2018)
18. Mirman, D.: Growth curve analysis and visualization using R. CRC press (2016)
19. de Oliveira, J., Salem, F., de Oliveira, E.H.T., Oliveira, D.B.F., de Carvalho, L.S.G., Pereira, F.D.: Os estudantes leem as mensagens de feedback estendido exibidas em juízes online? In: Anais do XXXI Simpósio Brasileiro de Informática na Educação. pp. 1723–1732. SBC (2020)

20. Pereira, F.D., Junior, H.B., Rodriguez, L., Toda, A., Oliveira, E.H., Cristea, A.I., Oliveira, D.B., Carvalho, L.S., Fonseca, S.C., Alamri, A., et al.: A recommender system based on effort: Towards minimising negative affects and maximising achievement in cs1 learning. In: International Conference on Intelligent Tutoring Systems. pp. 466–480. Springer (2021)
21. Pereira, F.D., Oliveira, E.H., Oliveira, D.B., Cristea, A.I., Carvalho, L.S., Fonseca, S.C., Toda, A., Isotani, S.: Using learning analytics in the amazonas: understanding students' behaviour in introductory programming. British journal of educational technology **51**(4), 955–972 (2020)
22. Pereira, F.D., Fonseca, S.C., Oliveira, E.H., Cristea, A.I., Bellhäuser, H., Rodrigues, L., Oliveira, D.B., Isotani, S., Carvalho, L.S.: Explaining individual and collective programming students' behavior by interpreting a black-box predictive model. IEEE Access **9**, 117097–117119 (2021)
23. Pereira, F.D., Fonseca, S.C., Wiktor, S., Oliveira, D.B., Cristea, A.I., Benedict, A., Fallahian, M., Dorodchi, M., Carvalho, L.S., Mello, R.F., et al.: Toward supporting cs1 instructors and learners with fine-grained topic detection in online judges. IEEE Access **11**, 22513–22525 (2023)
24. Pereira, F.D., Rodrigues, L., Henklain, M., Freitas, H., Oliveira, D.F., Cristea, A.I., Carvalho, L., Isotani, S., Benedict, A., Dorodchi, M., Oliveira, E.H.T.: Towards human-ai collaboration: a recommender system to support cs1 instructors to select problems for assignments and exams. IEEE Transactions on Learning Technologies pp. 1–14 (2022). https://doi.org/10.1109/TLT.2022.3224121
25. Pereira, F.D., de Souza, L.M., de Oliveira, E.H.T., de Oliveira, D.B.F., de Carvalho, L.S.G.: Predição de desempenho em ambientes computacionais para turmas de programação: um mapeamento sistemático da literatura. In: Anais do XXXI Simpósio Brasileiro de Informática na Educação. pp. 1673–1682. SBC (2020)
26. Quille, K., Bergin, S.: Cs1: how will they do? how can we help? a decade of research and practice. Computer Science Education **29**(2-3), 254–282 (2019)
27. Robins, A.V.: Novice programmers and introductory programming. In: The Cambridge Handbook of Computing Education Research, chap. 12, pp. 327–376. Cambridge University Press, Cambridge (2019)
28. Saito, T., Watanobe, Y.: Learning path recommendation system for programming education based on neural networks. International Journal of Distance Education Technologies (IJDET) **18**(1), 36–64 (2020)
29. Schwartz, B.M., Gurung, R.A.: Evidence-based teaching for higher education. American Psychological Association (2012)
30. Wasik, S., Antczak, M., Badura, J., Laskowski, A., Sternal, T.: A survey on online judge systems and their applications. ACM Computing Surveys (CSUR) **51**(1), 3 (2018)
31. Wilcox, R.R.: Introduction to robust estimation and hypothesis testing. Academic press (2011)
32. Yera, R., Martínez, L.: A recommendation approach for programming online judges supported by data preprocessing techniques. Applied Intelligence **47**(2), 277–290 (2017)
33. Zhao, W.X., Zhang, W., He, Y., Xie, X., Wen, J.R.: Automatically learning topics and difficulty levels of problems in online judge systems. ACM Transactions on Information Systems (TOIS) **36**(3), 27 (2018)
34. Zhou, W., Pan, Y., Zhou, Y., Sun, G.: The framework of a new online judge system for programming education. In: Proceedings of ACM Turing Celebration Conference-China. pp. 9–14. ACM (2018)