Automatic creating variation of CS1 assignments and exams

Filipe Dwan Pereira
Hermino Júnior
Federal University of Roraima, Boa Vista, BR
filipe.dwan@ufrr.br

Elaine H. T. Oliveira
Leandro S. G. Carvalho
David B. F. Oliveira
Federal University of Amazonas, Manaus, BR

Aileen Benedict
Mohsen Dorodchi
University of North Carolina at Charlotte, Charlotte, US

Alexandra I. Cristea
Durham University, Durham, UK

The adoption of Online Judge (OJ) environments by CS1 instructors has increased over the last few years [8–11, 14, 17, 19, 22, 24, 27–31]. OJs reduce instructors' workload in correcting learners' codes and provide instantaneous and accurate feedback to students about the correctness of their solutions [3, 6, 16, 21, 23, 26, 28]. Despite the benefits, there are still repetitive and laborious tasks to feed OJ systems. For example, the literature [1, 5, 13, 16, 18] recommends that instructors create variations of assignments and exams for different CS1 classes during the semesters to hamper plagiarism practice. By creating variations of assignments and exams, it is more difficult for students to use code solutions from past courses [13]. Indeed, an even more rigorous way of avoiding plagiarism would be to create personalized assignments and exams proactively for each student [5]. However, doing this manually is impractical, especially in classes with a high number of students.

To address this, we intend to create a mechanism for automatically selecting problems to compose new assignments and exams so that the new selection of problems is similar enough to the old in terms of problem topics and challenge levels. Ordinarily, the questions from an assignment available in CS1 courses share the same topic (e.g., conditional structure) [4, 12], and are scaffolded from easier to more challenging problems [7, 15]. In this work, we propose a way to generate N new assignments based on a previous one, called the "master assignment." These new assignments will then be composed of problems unique to the master assignment but similar in terms of topics and difficulty levels. Additionally, the same reasoning must be used to create new exams.

To accomplish our goal, we propose the procedure illustrated in Algorithm 1. In the procedure *getNewList*, $L = \{q_1 ... q_m\}$ represents a given master assignment, where m is the number of questions in L. The output $L' = \{q'_1 ... q'_m\}$ depicts a new assignment which has the same topic of L and requires a resolution effort (i.e., challenge level) similar to that of L. To create more than one new L's, we can manipulate the global variable K. For example, to create a second L' using a given L as input, we just need to assign 2 to the global variable K ($K \leftarrow 2$ in line 1 of the algorithm).

Notice that our procedure uses two auxiliary functions: *getTopic* and *findKthNearestNeighbour*. In the *getTopic* function, each question from the OJ we will have the tuple *(q, p)*, where q is the statement of the problem and p is a vector that represents the effort required to solve the question q. Each pair (q, p) has a topic t associated. The possible topics of the questions is based on the CS1 curriculum: *Sequential*, *Composite conditional structures*, *Chained conditional structures*, *Repeating structures by condition*, *Repeating structures by count*, *Vectors and Strings* and *Matrices*. As the questions of many OJs are not annotated with the topic of the question, the function *getTopic* uses machine learning and natural language processing techniques to predict the topic t of the statement q. More specifically, we will use a word embedding layer representation of each question q in a deep learning model, similar to what we have done in these works [2, 12, 25].

Algorithm 1 Creating new assignment/exam

```
1:  global const K ← 1                    ▷ K sets the ith L′ created based on L.
2:  procedure GETNEWLIST(L)
3:      L′ ← {}
4:      for (q, p) ∈ L do
5:          t ← getTopic(q)
6:          k ← K          ▷ Kth nearest neighbour of p is first used as p′
7:          q′ ← findKthNearstNeighbour(p, t, k)
8:          while q′ ∈ L′ do
9:              k ← k + 1
10:             q′ ← findKthNearstNeighbour(p, t, k)
11:         end while
12:         L′ ← q′ ∪ L′
13:     end for
14:     return L′                          ▷ new assignment/exam L′
15: end procedure
```

To find a problem that requires similar effort, *findKthNearstNeighbour* is used. Here, we use the nearest neighbour technique over the features, further discussed in previous works where we proposed and validated features to measure the students' required effort per problem [20, 23, 24]. The features will be the dimensions of the vector p that represents the effort required to solve q. In total, there are 21 features. Given a pair *(q, p)*, the vector p has the aggregation of the features' values based on the learners who solved that question q. To illustrate, given a question q_a, there is a feature called *loc* which is the lines of code a student used in their solution for question q_a. Thus, one of the dimensions of the vector p_a will be the average loc_{q_a} for all students who submitted accepted solutions for q_a.

Finally, we can assume that the questions from L' is sorted by difficult level. The reason is that L' is created based on the master list L, which has been previously sorted by difficult level by an instructor. In line 12 of Algorithm 1, the question q' in inserted in L' in the same interaction of the for loop (line 4) when $q \in L$ is accessed. That is, as the pair of questions (q, q') are potentially from the same topic and requires a similar effort to be solved, hence, L and L' are arranged in the same order.

EduComp'21, Abril 27–30, 2021, Jataí, Goiás, Brasil (On-line)

ACKNOWLEDGMENTS

This research, carried out within the scope of the Samsung-UFAM Project for Education and Research (SUPER), according to Article 48 of Decree nº 6.008/2006 (SUFRAMA), was partially funded by Samsung Electronics of Amazonia Ltda., under the terms of Federal Law nº 8.387/1991, through agreements 001/2020 and 003/2019, signed with Federal University of Amazonas and FAEPI, Brazil. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and the Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq grant 308513/2020-7).

REFERENCES

[1] Ibrahim Albluwi. 2019. Plagiarism in programming assessments: a systematic review. *ACM Transactions on Computing Education (TOCE)* 20, 1 (2019), 1–28.

[2] Tahani Aljohani, Filipe Dwan Pereira, Alexandra I Cristea, and Elaine Oliveira. 2020. Prediction of Users' Professional Profile in MOOCs Only by Utilising Learners' Written Texts. In *International Conference on Intelligent Tutoring Systems*. Springer, 163–173.

[3] Joe Michael Allen and Frank Vahid. 2021. Concise Graphical Representations of Student Effort on Weekly Many Small Programs. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 349–354.

[4] Ada Araujo, Daniel Lopes Zordan Filho, Elaine Harada Teixeira de Oliveira, Leandro Silva Galvão de Carvalho, Filipe Dwan Pereira, and David Braga Fernandes de Oliveira. 2021. Mapeamento e análise empírica de misconceptions comuns em avaliações de introdução à programação. In *Anais do Simpósio Brasileiro de Educação em Computação*. SBC, 123–131.

[5] Steven Bradley. 2016. Managing plagiarism in programming assignments with blended assessment and randomisation. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*. 21–30.

[6] Hermino Barbosa de Freitas Júnior, Filipe Dwan Pereira, Elaine Harada Teixeira de Oliveira, David Braga Fernandes de Oliveira, and Leandro Silva Galvão de Carvalho. 2020. Recomendação Automática de Problemas em Juízes Online Usando Processamento de Linguagem Natural e Análise Dirigida aos Dados. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 1152–1161.

[7] Marcos Avner Pimenta de Lima Lima, Leandro Silva Galvão de Carvalho, Elaine Harada Teixeira de Oliveira, David Braga Fernandes de Oliveira, and Filipe Dwan Pereira. 2021. Uso de atributos de código para classificação da facilidade de questões de codificação. In *Anais do Simpósio Brasileiro de Educação em Computação*. SBC, 113–122.

[8] Joseph de Oliveira, Felipe Salem, Elaine Harada Teixeira de Oliveira, David Braga Fernandes Oliveira, Leandro Silva Galvão de Carvalho, and Filipe Dwan Pereira. 2020. Os estudantes leem as mensagens de feedback estendido exibidas em juízes online?. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 1723–1732.

[9] Aracele Garcia de Oliveira Fassbinder, Tiago Gonçalves Gonçalves Botelho, Ricardo José Martins, and Ellen Francine Barbosa. 2015. Applying flipped classroom and problem-based learning in a CS1 course. In *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–7.

[10] Ingrid Lima dos Santos, David Braga Fernandes Oliveira, Leandro Silva Galvão de Carvalho, Filipe Dwan Pereira, and Elaine Harada Teixeira de Oliveira. 2020. Tempos de Transição em Estados de Corretude e Erro como Indicadores de Desempenho em Juízes Online. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 1283–1292.

[11] Filipe Dwan, Elaine Oliveira, and David Fernandes. 2017. Predição de zona de aprendizagem de alunos de introdução à programação em ambientes de correção automática de código. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Vol. 28. 1507.

[12] Samuel C Fonseca, Filipe Dwan Pereira, Elaine HT Oliveira, David BF Oliveira, Leandro SG Carvalho, and Alexandra I Cristea. 2020. Automatic Subject-based Contextualisation of Programming Assignment Lists. EDM.

[13] Max Fowler and Craig Zilles. 2021. Superficial Code-guise: Investigating the Impact of Surface Feature Changes on Students' Programming Question Scores. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 3–9.

[14] Rodrigo Elias Francisco and Ana Paula Ambrosio. 2015. Mining an Online Judge System to Support Introductory Computer Programming Teaching.. In *EDM (Workshops)*. Citeseer.

[15] Marcos Lima, Leandro Silva Galvão de Carvalho, Elaine Harada Teixeira de Oliveira, David Braga Fernandes Oliveira, and Filipe Dwan Pereira. 2020. Classificação de dificuldade de questões de programação com base em métricas de código. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 1323–1332.

[16] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 55–106.

[17] Filipe Pereira, Elaine Oliveira, David Fernandes, Hermino Junior, and Leandro Silva Galvão de Carvalho. 2019. Otimização e automação da predição precoce do desempenho de alunos que utilizam juízes online: uma abordagem com algoritmo genético. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, Vol. 30. 1451.

[18] Filipe Dwan Pereira, Linnik Maciel de Souza, Elaine Harada Teixeira de Oliveira, David Braga Fernandes de Oliveira, and Leandro Silva Galvão de Carvalho. 2020. 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*. SBC, 1673–1682.

[19] Filipe Dwan Pereira, Samuel C Fonseca, Elaine HT Oliveira, David BF Oliveira, Alexandra I Cristea, and Leandro SG Carvalho. 2020. Deep learning for early performance prediction of introductory programming students: a comparative and explanatory study. *Brazilian journal of computers in education.* 28 (2020), 723–749.

[20] Filipe D Pereira, Hermino Junior, Luiz Rodriguez, Armando Toda, Elaine HT Oliveira, Alexandra I Cristea, David Oliveira, Leandro Carvalho, Samuel Fonseca, Ahmed Alamri, and Seiji Isotani. 2021. A recommender system based on effort: towards minimising negative affects and maximising achievement in CS1 learning. In *International Conference on Intelligent Tutoring Systems*. Springer.

[21] Filipe D Pereira, Elaine Oliveira, Alexandra Cristea, David Fernandes, Luciano Silva, Gene Aguiar, Ahmed Alamri, and Mohammad Alshehri. 2019. Early dropout prediction for programming courses supported by online judges. In *International Conference on Artificial Intelligence in Education*. Springer, 67–72.

[22] Filipe Dwan Pereira, Elaine HT Oliveira, David Fernandes, and Alexandra Cristea. 2019. Early performance prediction for CS1 course students using a combination of machine learning and an evolutionary algorithm. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, Vol. 2161. IEEE, 183–184.

[23] Filipe Dwan Pereira, Elaine HT Oliveira, David Oliveira, Alexandra I Cristea, Leandro Carvalho, Samuel Fonseca, Armando Toda, and Seiji Isotani. 2020. Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British journal of educational technology.* (2020).

[24] Filipe Dwan Pereira, Elaine H T Oliveira, and David F B Oliveira. 2018. *Uso de um método preditivo para inferir a zona de aprendizagem de alunos de programação em um ambiente de correção automática de código.* Mestrado em Informática. Universidade Federal do Amazonas, Manaus.

[25] Filipe Dwan Pereira, Francisco Pires, Samuel C Fonseca, Elaine HT Oliveira, Leandro SG Carvalho, David BF Oliveira, and Alexandra I. 2021. Towards a Human-AI hybrid system for categorising programming problems *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432422

[26] Filipe D Pereira, Armando Toda, Elaine HT Oliveira, Alexandra I Cristea, Seiji Isotani, Dion Laranjeira, Adriano Almeida, and Jonas Mendonça. 2020. Can we use gamification to predict students' performance? A case study supported by an online judge. In *International Conference on Intelligent Tutoring Systems*. Springer, 259–269.

[27] Tomohiro Saito and Yutaka Watanobe. 2020. Learning Path Recommendation System for Programming Education based on Neural Networks. *International Journal of Distance Education Technologies (IJDET)* 18, 1 (2020), 36–64.

[28] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2018. A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)* 51, 1 (2018), 3.

[29] Ruiguo Yu, Zhiyong Cai, Xiuping Du, Muwen He, Zan Wang, Binlan Yang, and Peng Chang. 2015. The research of the recommendation algorithm in online learning. *International Journal of Multimedia and Ubiquitous Engineering* 10, 4 (2015), 71–80.

[30] Wayne Xin Zhao, Wenhui Zhang, Yulan He, Xing Xie, and Ji-Rong Wen. 2018. Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Transactions on Information Systems (TOIS)* 36, 3 (2018), 27.

[31] Daniel Lopes Zordan Filho, Elaine Harada Teixeira de Oliveira, Leandro Silva Galvão de Carvalho, Marcela Pessoa, Filipe Dwan Pereira, and David Braga Fernandes de Oliveira. 2020. Uma análise orientada a dados para avaliar o impacto da gamificação de um juiz on-line no desempenho de estudantes. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. SBC, 491–500.