Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 198 (2022) 391–396

Procedia Computer Science

www.elsevier.com/locate/procedia

The 2nd International Workshop on Artificial Intelligence for Natural Language Processing (IA&NLP'21)
November 1-4, 2021, Leuven, Belgium

Comparative study between automatic hint generation approaches in Intelligent Programming Tutors

Mariam MAHDAOUI[1,*], Said NOUH[1], My Seddiq ELKASMI ALAOUI[2],Mounir SADIQ[1]

1 LTIM, Hassan II University, Casablanca, Morocco
[2] LIS, Hassan II University , Casablanca, Morocco

Abstract

In all engineering discipline in universities, introductory programming is an essential part of the curriculum. However, many students find great difficulty with the learning of programming without instructor assistance and it becomes a barrier to their further studies of computer science and other disciplines.
Intelligent Programming Tutors (IPTs) assist students to overcome difficulties in solving programming exercises, when they don't know how to proceed or they unable to find an error in their current state. Such a student may benefit from hints, guiding her toward a more complete and/or more correct version and allowing her to continue working on her own (Aleven et al., 2016). However, traditional IPTs take much time to create the expert knowledge because in computer programming tasks, the space of possible solutions is too large (Piech et al., 2015b). New development methods have emerged in recent years, which provide hints without an expert model. These techniques are based on data-driven approaches and/or program repair techniques and provide hints in the form of edits (such as inserting or deleting a piece of code). Students can apply these hints in their current state to change it into a more correct and/or more complete state. This paper aims to provide a best understanding of these new techniques and presents a comparative study between different approaches of automatic hint generation by presenting the forces and weaknesses of each approach.

Keywords: Hint generation, automatic feedback; intelligent programming tutors; data-driven approach; programming education

* Corresponding author. Tel.:+212663807817;
E-mail address: mahdaoui.mariam@gmail.com

1. Introduction

Today, knowledge is an essential lever for any economic and social development. Universities and training centers have noticed that a certain number of learners encounter obvious difficulties in their learning of programming, which can become an obstacle to the success of their training. (Winslow, 1996) [1], (Jenkins, 2002) [2], (Robins, Rountree & Rountree, 2003) [3], (Lahtinen, Ala-Mutka, & Järvinen, 2005) [4], (Gomes & Mendes, 2007) [5], (Tan, Ting, & Ling, 2009) [6], (Luxton-Reilly et al., 2018) [7].

According to the ACM/AIS/IEEE report, "Universities report that regularly 50% or more of their students who initially chose computer science studies quickly decide to drop out" (ACM, AIS, & IEEE, 2005) [8]. For his part, Winslow, points out that it takes about 10 years to turn a novice into a programming expert (Winslow, 1996) [1]. In the same way (Mead et al., 2006) [9] point out that "over the past 25 years, national and international studies have provided empirical indicators that learning programming is indeed difficult for most students".

Thus, our work falls within the field of technology for Computing Environment for Human Learning (CEHL), which, according to (Tchounikine, 2009) [10], refers us to work, the objective of which is to develop concepts, methods and techniques to implement an Intelligent Programming Tutor (IPT) that facilitates or reinforce human learning. Based on the feedback to help the learner to complete an exercise or a course in programming in case of difficulty.

Research has shown feedback is an essential component of the learning process. Providing feedback on programming assignments is an integral part of a class on introductory programming and requires substantial effort by the teaching personnel.

Supporting students' learning is a fundamental challenge in computer science (CS) education in particular, providing personalized feedback to a large number of students.

In order to address this problem, researches have designed Intelligent Programming Tutors that help students learn solving programming exercises.

Current trends in the IPTs are to use data-driven techniques [11- 17] and use technical capabilities of program repair to give hints to students [18, 19, 20] by offering a personalized suggestion for how to progress or fix an error [21,19, 28].

2. Data-driven Tutoring

Data-driven tutoring and automatic programming hint generation have grown in recent years due to the surge in popularity of Massive Open Online Courses (MOOCs) (Pappano, 2012) [32]; the biggest challenge in such a setting is scaling personalized feedback to any number of students.

Automatic programming hint is any type of feedback that support students to complete a programming task where no human intervention is required when the hint is requested. These programming hints can help them to identify errors in their code, suggest potential ways to proceed, recommend concepts to revise or clarify the task requirements [30].

These are called data-driven tutors because they generate hints from prior students' data [21,16,17,13], allowing them to support diverse solutions [13,24] and scale to support any number of a large number of students with little instructor effort.

3. Data-driven hint generation techniques

Recently, many approaches to generating automated programming hints have been proposed. These approaches are based on a different ideas and techniques, such as machine learning, the utilization of teacher or peer data, debugging technique, and other methods. We present here some of them.

3.1. Hint Factory

A data-driven hint generation approach was used first in the Hint Factory, a hint-generation method used in a logic proof tutoring system (Barnes & Stamper, 2008) [29] and later adapted to the domain of programming [24].

To produce next-step hints for students the Hint Factory technique uses peer data and a Markov Decision Process (MDP): Program submissions made by peers are transformed into a state space where each state represents a particular class of programs, and edges connecting the states represent how students transition between them (thereby indicating paths to various solutions). A student's submission is then matched to a state in this state space, and the MDP is used to determine the best next state for that student [30].

3.2. The Target Recognition – Edit Recommendation :

In the context of computer programming, many data mining techniques cannot be applied because computer programs are represented in the form of abstract syntax trees (AST). Several works have suggested representing the AST by the subtree patterns it contains (Nguyen et al., 2014; Zhi et al., 2018). Such representations are used in natural language processing domain, where texts are represented as a collection of the words or n-grams they contain (Broder et al., 1997).

The Target Recognition – Edit Recommendation (TR-ER) algorithm (Zimmerman and Rupakheti 2015) represents the AST as a multiset of all of its subtrees of a specific shape (called pq-Grams).

The algorithm defines the target state as the closest correct solution to the student's current code, calculated by the pq-gram distance (Augsten et al. 2005), then suggests hints to insert delete or replace code based on the missing or extra pq-grams in the student's code.

3.3. Ast2vec

Ast2vec(B. Paaßen, 2021) [31], a neural network that encodes the students current abstract syntax tree as a vector, computes the difference in the encoding space to the most common correct solution, applies a linear transformation to that difference, adds the result to the student's current position, and decodes the resulting vector to achieve the next-step prediction.

3.4. ITAP

ITAP (Intelligent Teaching Assistant for Programming) [13] is a data-driven tutor for programming in Python (Rivers and Koedinger 2017) generates hints using this process:

1. Canonicalize and normalization of all code by removing non-important syntactic variations.
2. Identify the closest correct, submitted solution to the student's current code.
3. Apply path construction to identify any closer, correct solutions.
4. Identify a target state on the path to the solution that will make a good macro-level hint, as defined by a custom desirability metric.
5. Extract a single edit to present to the student as a text hint.

3.5. The Continuous Hint Factory (CHF)

The Continuous Hint Factory (CHF) (Paaßen et al. 2018) attempts to define the target state as the next step that an average, capable student would have taken.

First, it computes the distance between the student's trace (including code history) and the traces of all students in the training dataset who got closer to a correct solution, using a dynamic time warping (DTW) (Vintsyuk 1968) approach.

Second, the CHF defines the target state based on predictions of how these capable students would proceed in the student's situation, using Gaussian Process Regression on the DTW distances (Paaßen et al. 2017). The CHF identifies tree edits, which bring the student closer to the target state, using a tree grammar to select only valid edits for a given programming language.

3.6. The SourceCheck algorithm

The SourceCheck algorithm (Price et al. 2017d) [23] matches an incorrect student program to the closest known solution based on a distance measure defined by the authors. It calculates this using a code-specific distance metric, which, can match common code elements from the two ASTs, even when they appear in different orders or locations, unlike tree edit distance. The edits needed to convert the student's program to this solution are then presented as hints.

3.7. Clustering techniques

As known, the space of programs for the same programming task is very large and it is impossible to design feedback for all possible programs. So, several researchers try to group programs into clusters, for which similar feedback can be given (Choudhury et al., 2016; Glassman et al., 2015; Gross et al., 2014; Gulwani et al., 2018). When a student needs a hint on how to change his code, one can simply check which cluster the student's program belongs to, and assign the feedback for that cluster.

Several researches used different clustering strategies: (Choudhury et al., 2016; Gross et al., 2014; Zhang and Shasha, 1989) are grouping programs by calculating the tree edit distance whereas (Gulwani et al., 2018) are grouping programs by their control flow and (McBroom et al., 2021) are grouping programs by the unit tests they pass.

3.8. Program repair techniques:

Automated program repair (APR) is an emerging new technology that has recently been actively researched. An APR system fixes software bugs automatically, only requiring a test suite that can drive the repair process. Failing tests in the test suite become passing after repair, which manifests as a bug fix. Generating feedback from program repair is an active area of research:

Yi et al. [25] explore diverse automated program repair (APR) systems to generate feedback in Intelligent Programming Systems. Gulwani [26] use the existing correct student solutions, to provide a complete repair of an incorrect student attempts and generates a feedback as a textual description of the generated repair very similar to the feedback generated by AutoGrader [27].

AutoGrader [27] uses a reference solution and a set of potential corrections in the form of expression rewrite rules provided by the course instructor and searches for a set of minimal corrections using program synthesis to generate feedback.

SYNFIX [45] automatically correct syntactic errors in student programs, by using peer data to train a recurrent neural network (RNN) to predict correct sequences of program pieces (tokens). When a student need help, the RNN can then be used to find new or alternative tokens to insert near the error location in order to correct the program. Corrections found are used to produce hints for the student.

4. Hint generation framework

A recent research made by (J.McBroom, 2019) [30] presents the Hint Iteration by Narrow-down and Transformation Steps (HINTS) framework and demonstrate that all these hint approaches are built up from just two simple operations applied iteratively . These steps involved narrowing down and transforming hints from earlier levels.

Each approach need a pool of data as input (hint data) used to produce hints. To produce new hint data, the input is processed by 1) transformations: changing the data representation and/or 2) narrowing down: selecting a subset of this data based on a relevance criterion and/or a quality criterion as shown in the Table1.

Table 1. Summary of Hint generation techniques.

Hint generation technique	Input data	transformation step	Narrow down level: how to select the best next state
Hint Factory	Peer data, student program	AST	MDP
TR-ER	Peer data, student program	AST, Tree Patterns	pq-Gram distance
Ast2vec	Peer data, student program	AST, Euclidian vector	weighted sum of peer edits
ITAP	Correct peer data, teacher data, student program	AST,canonical form	Tree edit distance
CHF	Peer solutions, student program	AST, tree grammar	DTW distance, Gaussian Process Regression
Source Check	Peer solutions, student program	graph with nodes	custom distance measure
SYNFIX	Peer data, student program	sequences of tokens	RNN

5. CONCLUSION

This paper presents the existing techniques for automatically generating data-driven feedback for students working on code writing programming problems. However, because they are generated from data, these hints are not always benefit for the students, and if so it only suggest how to progress, without the expert-authored explanations and domain principles found in many tutoring systems. Techniques could be improved by a method of selecting the most useful edit to present, and improving feedback with more explanations.

In addition, many data mining techniques cannot be applied because they don’t support syntax trees as input. Representing computer programs into vectors in Euclidean space is an interesting new research field because it allows the exploitation of at least a hundred data mining techniques for the future work.

Acknowledgements

This work is carried out as part of the research project entitled: “A mutualized platform of manipulation of TP and scientific manipulation of research for the faculty of sciences Ben M'Sik”

References

[1] Winslow, L. E. (1996). Programming pedagogical psychological overview. ACM SIGCSE Bulletin, 28(3), 17–22. https://doi.org/10.1145/234867.234872

[2] Jenkins, T. (2002). On The Difficulty of Learning to Program. In Proceedings of the 3rdAnnual Conference of the LTSN Centre for information and Computer Sciences. (pp. 53–58). Retrieved from http://www.psy.gla.ac.uk/~steve/localed/jenkins.html

[3] Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2), 137–172.

[4] Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. ACM SIGCSE Bulletin, 37(3), 14–18.https://doi.org/10.1145/1151954.1067453

[5] Gomes, A., & Mendes, a. J. (2007). Learning to program - difficulties and solutions. In International Conference on Engineering Education – ICEE.

[6] Tan, P. H., Ting, C. Y., & Ling, S. W. (2009). Learning difficulties in programming courses: Undergraduates’ perspective and perception. In IEEE (Ed.), ICCTD 2009 – International Conference on Computer Technology and Development (pp. 42–46).https://doi.org/10.1109/ICCTD.2009.188

[7] Luxton-Reilly, A., Simon, Beker, B. A., Albluwi, I., Giannakos, M., Amruth N., K., Szabo, C. (2018). 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).

[8] ACM, AIS, & IEEE. (2005). Computing Curricula 2005, the overview Report. New York.

[9] Mead, J., Simon, G., John, H., Richard, J., Juha, S., Caroline, S. C., & Linda, T. (2006). A Cognitive Approach to Identifying Measurable Milestones for Programming Skill Acquisition. ACM SIGCSE Bulletin, 38(4), 182–194.

[10] Tchounikine, P. (2009). Précis de recherche en ingénierie des EIAH. Retrieved October 20, 2016, from http://membres-liglab.imag.fr/tchounikine/Precis.html

[11] Jin, Wei, T. Barnes, J. Stamper, M. J. Eagle, Matthew W. Johnson, and L. Lehmann. "Program representation for automatic hint generation for a data-driven novice programming tutor". International Conference on Intelligent Tutoring Systems, 2012, pp. 304-309. https://doi.org/10.1007/978-3-642-30950-2_40

[12] L. Timotej and I. Bratko. "Data-driven program synthesis for hint generation in programming tutors". International Conference on Intelligent Tutoring Systems, 2014, pp. 306-311.

[13] K. Rivers and K. Koedinger. "Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor". International Journal of Artificial Intelligence in Education, vol. 27, no. 1, pp. 37-64, 2015. https://doi.org/10.1007/s40593-015- 0070-z

[14] P. Thomas and T. Barnes. "Creating data-driven feedback for novices in goal-driven programming projects". International Conference on Artificial Intelligence in Education, 2015, pp. 856-859.

[15] P. Thomas and T. Barnes. "An Exploration of Data-Driven Hint Generation in an Open- Ended Programming Problem". Educational Data Mining (Workshops), 2015.

[16] P. Chris, M. Sahami, J. Huang and L. Guibas. "Autonomously generating hints by inferring problem solving policies". Proceedings of the Second ACM Conference on Learning@ Scale, 2015, pp. 195-204.

[17] G. Alex, B. Heeren, J. Jeuring and L. Binsbergen. "Ask-Elle: an adaptable programming tutor for Haskell giving automated feedback". International Journal of Artificial Intelligence in Education, pp. 1-36, 2016.

[18] Loris D'Antoni, Roopsha Samanta, and Rishabh Singh. 2016. Qlose: Program Repair with Quantitative Objectives. In Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada,July 17-23, 2016, Proceedings, Part II. 383–401. http://dx.doi.org/10.1007/978-3-319-41540-6_21

[19] Reudismam Rolim, Gustavo Soares, Loris D'Antoni, Oleksandr Polozov, Sumit Gulwani, Rohit Gheyi, Ryo Suzuki, and Björn Hartmann. 2017. Learning Syntactic Program Transformations from Examples. In Proceedings of the 39th International Conference on Software Engineering (ICSE '17). IEEE Press, Piscataway, NJ, USA, 404–415. https://doi.org/10.1109/ICSE.2017.44

[20]. Price, T.W., Dong, Y., Lipovac, D.: iSnap: towards intelligent tutoring in novice programming environments. In: Proceedings of the ACM Technical Symposium on Computer Science Education, pp. 483–488 (2017)

[21] Choudhury, R.R., Yin, H., Fox, A.: Scale-driven automatic hint generation for coding style. In: Proceedings of the International Conference on Intelligent Tutoring Systems, pp. 122–132 (2016)

[22]. Piech, C., Sahami, M., Huang, J., Guibas, L.: Autonomously generating hints by inferring problem solving policies. In: Proceedings of the second (2015) ACM Conference on Learning@ Scale, pp. 195–204 (2015)

[23]. Price, T., Zhi, R., Barnes, T.: Evaluation of a data-driven feedback algorithm for open-ended programming. In: International Educational Data Mining Society (2017)

[24] Price, T.W., Dong, Y., Barnes, T.: Generating data-driven hints for open-ended programming. In: Proceedings of the International Conference on Educational Data Mining (2016)

[25] Jooyong Yi, Umair Z. Ahmed, Amey Karkare, Shin Hwei Tan, and Abhik Roychoudhury. 2017. A Feasibility Study of Using Automated Program Repair for Introductory Programming Assignments. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017). ACM, New York, NY, USA, 740–751. https://doi.org/10.1145/3106237.3106262

[26] Sumit Gulwani, Ivan Radiček, and Florian Zuleger. 2018. Automated Clustering and Program Repair for Introductory Programming Assignments. CoRR abs/1603.03165 (2018). arXiv:1603.03165 http://arxiv.org/abs/1603.03165

[27] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. 2013. Automated Feedback Generation for Introductory Programming Assignments. In Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13). ACM, New York, NY, USA, 15–26. https://doi.org/10.1145/2491956.2462195

[28] S. Bhatia and R. Singh, "Automated correction for syntax errors in programming assignments using recurrent neural networks," arXiv preprint arXiv:1603.06129, 2016.

[29] J. Stamper, T. Barnes, L. Lehmann, and M. Croy, "The hint factory: Automatic generation of contextualized help for existing computer aided instruction," in Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track, pp. 71–78, 2008.

[30] J.McBroom, I.Koprinska, and K. Yacef "A Survey of Automated Programming Hint Generation - The HINTS Framework" arXiv:1908. 11566v1 [cs.HC] 30 Aug 2019

[31] B. Paaßen, J. McBroom, B. Je_ries, I. Koprinska, and K. Yacef. ast2vec: Utilizing recursive neural encodings of python programs. Journal of Educational Datamining, 2021. in press.

[32] Pappano, L. (2012). The Year of the MOOC. The New York Times, 2(12), 2012