UDC 004.89 DOI: https://doi.org/10.30837/IYF.IIS.2024.257

INTELLIGENT ANALYSIS USAGE IN AUTOMATED GRADING SYSTEMS

Seliutin D. A.
Scientific adviser – Candidate of Science, Associate Professor, Yashyna O. S.
National Aerospace University «Kharkiv Aviation Institute», Department of Computer Sciences and Information Technologies of the Faculty of Aircraft Management Systems, Kharkiv, Ukraine
e-mail: denisselutinds@gmail.com

Technology integration in education has grown in a more important role during the COVID-19 epidemic. As standard manual grading techniques have been shown to be insufficient – automation grading provides a more effective approach. However, assessing programming activities is challenging because students employ a variety of coding styles and technical stacks. Manual evaluation offers flexibility but lacks consistency and scalability. Furthermore, advanced levels of programming skill requires completely new approaches to task assessment. To solve these issues, improved approaches including both dynamic and static code analysis, reinforced by machine learning techniques, are essential.

Technology integration in the dynamic educational landscape has transformed teaching, learning, and data assessment. Automating workout grading technology is a major advancement and innovation that takes place during COVID-19. Traditional manual grading methods sometimes are inadequate for current education. Automation offers a more efficient solution and allows for personalized learning and innovative teaching methods.

The variety of task solutions, coding methodologies, and technology stacks make programming language learning manual assessment less strict in offering comments. Meanwhile, manual assessment depends heavily on many human variables in the educational process, such as manual verification by teachers for task feedback and the use of different technological stacks that may result in the rejection of assignments.

This leads to being essential for students to have the freedom to explore different approaches and not be limited by predefined technology stacks or frameworks set by teachers when learning programming languages, data analysis, or AI-related courses. This allows them to reach their goals independently or at least understand the boundaries they need to work within.

Nevertheless, automation of task assessment is a challenging issue because of the complexity of programming code and the diverse approaches that can be used to perform a task or hindering automation systems from providing accurate feedback [1]. The challenges are connected to a way of assessing a written code as a part of programming exercises because code can be provided as an entire program or just a section of it.

The idea that students have produced code and merely need to add a missing component [2] simplifies assessment but reduces instructional and practical value. This method is sufficient for learning programming language syntax, vocabulary, and basics. How about advanced levels where students write contract-based code? The solution to this complicated verification is unknown due to several limitations and can be named as advance code task assessment.

Advanced code task assessment needs test coverage and code analysis [1]. Heuristics or intellectual code analysis can help identify the core cause of execution errors and logical errors that caused assessment failures when dynamic and static code analysis simply found quality code metrics. This type of intellectual code examination involves machine learning.

Code identification and breakdown using machine learning use categorization and natural language processing instead of static code parsing and interpretation. Research shows that RNN-based deep learning models are the most common way for this type of evaluation [3]. LSTM- and GRU-based models may achieve the same goal with less precision. These methods generate a vector representation of the code that can be customized using features and training. Vectors can identify error-causing, style-changing, and logically challenging code. [4].

To comprehend the components, examine them next. This knowledge can be compared to the teacher's expected structure or other task execution frameworks. This helps create and distribute a student's most important resource: a coding problem-solving guide. This method improves automated evaluation tools by using intelligence analysis with neural networks to discover code flaws and give students useful feedback without teacher intervention.

The research conducted suggests enhancing current code assessment methods by intelligent analysis with deep learning. This enables development of flexible and responsive programming tools for learning.

References:

1. Ala-Mutka K. A survey of automated assessment approaches for programming assignments // Computer Science Education. 2005. Vol. 15, No. 2. P. 83–102.

2. Automatic Generation and Grading of Programming Exercises / Andy DuFrene. California Polytechnic State University, 2016. URL: https://core.ac.uk/display/77510984 (дата звернення: 23.02.2024).

3. Tushar S., Maria K., Stefanos G., Rohit T., Indira V., Hadi M., Federica S. A survey on machine learning techniques applied to source code // Journal of Systems and Software. 2024. Vol. 209.

4. Ahire P., Abraham J. Perceive Core Logical Blocks of a C Program Automatically for Source Code Transformations // Intelligent Systems Design and Applications. Cham, 2020. P. 386–400.