IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.002, LIIR082
DOI: https://doi.org/10.52731/liir.v002.082

Teacher-oriented Source Code Similarity Detection and Visualization for Programming Assignments

Maxim Mozgovoy [*], Evgeny Pyshkin [*], John Blake [*],
Marina Purgina [*], Agnes Leung [*]

Abstract

In programming classes, instructors need to work with numerous exercise submissions to verify whether the submitted source code meets the requirements, and whether there is any unauthorized borrowing of code fragments. The checking procedure is laborious requiring much unproductive effort and time. However, ignoring instances of potential plagiarism may negatively impact learner motivation. Despite the existence of practical tools developed for software testing and similarity detection, there are still issues in developing an open-source submission assessment system that would streamline the classroom workflow. This paper describes a practical submission assessment system that reduces the time teachers spend checking the solutions submitted by students.

Keywords: programming instruction, source code similarity, classroom workflow automation, similarity visualization.

1 Introduction

Educational institutions responded to societal lock-downs in 2020-22 with a significant increase in the use of digital and online teaching and learning platforms. Naturally, programming and software development-related classes are no exception in this extensive transformation of teaching practices involving using learning management systems (LMS) such as Moodle, online meeting tools, testing frameworks, version control, and bug tracking systems. What makes this transformation challenging specifically for software education is that software-related classes require many activities which involve high degrees of interactivity and collaboration [1][2].

Even when checking students' submissions using submission management automation provided by Moodle, much time is still spent on source code testing and similarity detection. Some available plagiarism detection tools (such as JPlag [3], MOSS [4], or Plaggie [5]) are very helpful but still require class instructor to complete a lot of manual operations.

Figure 1 displays the major use cases of a source code-based submission assessment process, which helped us to identify the features needed for a practical system. Although

[*] University of Aizu, Aizu-Wakamatsu, Japan

[12] M. Freire, "Visualizing program similarity in the Ac plagiarism detection system," in Proceedings of the Working Conference on Advanced Visual Interfaces, ser. AVI '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 404–407; doi:10.1145/1385569.1385644.

[13] M. Mozgovoy and E. Pyshkin, "Plagiarism Detection Systems for Programming Assignments: Practical Considerations," Proc. ICSEA 2020, IARIA, 2020, pp. 16-20.

[14] M. Novak, M. Joy, and D. Kermek, "Source-code similarity detection and detection tools used in academia: a systematic review," ACM Transactions on Computing Education (TOCE), vol. 19, no. 3, 2019, pp. 1–37; doi:10.1145/3313290.

[15] M. J. Wise, "String similarity via greedy string tiling and running Karp-Rabin matching," Online Preprint, Jan 1993, vol. 119, no. 1, 1993, pp. 1–17; https://www.researchgate.net/publication/262763983.

[16] H. Cheers, Y. Lin, and S. P. Smith, "Academic source code plagiarism detection by measuring program behavioral similarity," IEEE Access, vol. 9, 2021, pp. 50391–50412; doi:10.1109/ACCESS.2021.3069367.

[17] C. Jewitt, "Multimodality and literacy in school classrooms," Review of research in education, vol. 32, no. 1, 2008, pp. 241–267; doi:10.3102/0091732X07310586.

[18] M. Dressman, "Multimodality and language learning," The handbook of informal language learning, Wiley, 2019, pp. 39–55; doi:10.1002/9781119472384.ch3.