Copy-Past Culture: Examining the Causes and Solutions to Source Code Plagiarism

Jayden Manyrath
jaydenmichaelmanyrath@gmail.com
https://orcid.org/0009-0003-2678-1706

Kaleb Kirubel
kaleb.kirubel1@gmail.com
https://orcid.org/0009-0006-8426-8945

Thomas Cruz
tomasillo2567@gmail.com
https://orcid.org/0009-0004-0010-670X

Abstract

In an era marked by the increasing digitization of society, the issue of source code plagiarism has emerged as a persistent concern. This research paper delves into the problem of source code plagiarism within educational settings, exploring its implications, potential remedies, and the associated hurdles in implementing these solutions. Source code plagiarism involves the unauthorized copying of code without proper attribution, and it has been on the rise in educational institutions due to various contributing factors. This paper sheds light on the educational system's pressures, time constraints, lofty expectations, and the allure of quick completion that make source code plagiarism appealing to students. Furthermore, it highlights the lack of understanding among students regarding academic integrity and citation methods, exacerbating the problem. Source code plagiarism not only hampers students' intellectual development and problem-solving skills but also undermines the fairness of assessments, posing grading challenges for educators. Nevertheless, there are several potential solutions. While proactive methods focus on prevention through education and policy, reactive methods employ AI-driven plagiarism detectors for detection. However, these solutions are not without their challenges, such as the issue of false positives in plagiarism detection and the potential adversarial response from students. In conclusion, source code plagiarism is a growing problem in modern society that can not be avoided any longer. Potential solutions to source code plagiarism should be taken into account while considering their withdrawals. Computer science and programming courses should foster a sense of integrity to avoid source code plagiarism and develop new generations of coders for the future.

Keywords: artificial intelligence, education, plagiarism.

1. Introduction

In today's digital age, where technology underpins nearly every facet of modern life, programming and coding skills have become increasingly valuable. As a result, educational institutions' computer science and programming courses have seen a surge in enrollment. With this surge, we have also seen a rise in a more pressing issue - source code plagiarism. Many have claimed the work of others as their own to ensure the sanctity of their grades or reputations. While the motives of plagiarism range from incompetence to indolence, the consequences for their lack of diligence remain all the same. Source code plagiarism is among the most prevalent forms of plagiarism in this digital age. Detecting source code plagiarism has become challenging due to the large amount of code teachers have to detect manually. Additionally, new loopholes to avoid AI detection are being continuously created.

Previous research touches upon the basis of source code plagiarism and its effects. However, most of the research obtained focuses on a sole aspect of source code plagiarism while also not considering source code plagiarism in education. Nevertheless, while the sources used in this paper focus on the implication plagiarism can have on our society as a whole, our paper emphasizes an individualistic perspective to help others understand potential circumstances that could hinder a student's learning.

This paper aims to explore the multifaceted issue of source code plagiarism in educational settings, investigating its implications, potential remedies, and the challenges that arise when attempting to address this problem. In doing so, we will shed light on the reasons behind the increasing prevalence of source code plagiarism, the impact on students, educators, and the integrity of educational institutions, and the strategies employed to combat this issue.

2. Understanding the Problem

Source code plagiarism is the act of copying code from another source without proper attribution or authorization. "Plagiarism [has become] … a long-standing issue in academic institutions" due to the multiple ways students can go about plagiarizing others' source code (Cheers et al., 2021, p.50391). With the growing simplicity and security behind evading detection, "plagiarism can be a difficult and time-consuming task to identify, often requiring a large effort on the part of academics to review and assess assignment submissions for plagiarism" (Cheers et al., 2021, p.50391). This disarray has become increasingly concerning, as "overall employment in computer and information technology occupations is projected to grow 14.6% from 2021 to 2031" (Krutsch, 2022).

One method to plagiarize source code is the copy-and-paste method, where a student submits someone else's work unchanged without giving them proper credit. Regardless of how the information is presented, this form of blatant plagiarism clearly shows an individual's disregard for their education. Students often minimally alter the source code to differentiate it from its original work and prevent detection. In "extreme cases of plagiarism-hiding transformations," students would use "[p]ervasive plagiarism-hiding transformations" to alter the source code, "transforming it such that it bares little cosmetic or structural similarity" (Cheers et al., 2021, p.50392).

"In addition to verbatim copying of assignments between students, a programming assignment may also be considered plagiarized if code is reused between [a student's own] assignments (self plagiarizing)" (Buitendag et al., 2013, p.46). Unbeknownst to some students, reusing their old code to pass off as new work can be just as inappropriate as copying someone else's code because it deceives the reader of the true origin of the work.

Lastly, referencing someone else's code while incorrectly citing their work is also considered plagiarism. Although students do not attempt to take credit for original work that is not theirs, they misinform readers about whom the work belongs to, thus plagiarizing it. "This suggests that students' lack of understanding about plagiarism is not strongly correlated to a certain level of education…[as] many students think plagiarism can be avoided by citation and reference alone" (Buitendag et al., 2013, p.48).

3. The Root Causes

What exactly causes source code plagiarism in education? Source code plagiarism can be appealing to students in computer science and programming courses for several different reasons. Not only is there a plethora of accessible resources online, but it can also be seen as a quick and easy way to complete assignments for those who lack a general understanding of the topic, those who cannot correctly manage their time, and those who may crumble under high expectations from parents or teachers. "There are numerous examples of websites and services that host searchable code which is accessible by the public, e.g., question and answer sites that provide ready-made solutions to programming problems and websites where, for a relatively small fee, a programmer can be hired to complete a task. This provides enough resources to tempt a student to plagiarize part of or an entire assignment" (Buitendag et al., 2013, p.46). Source code plagiarism is often used as a coping mechanism for students who struggle with courses and fall behind.

Additionally, there are cases where source code plagiarism occurs unintentionally. "Plagiarism in student assignments continues to be a major concern in universities, mainly because of students' inadequate understanding of actions that constitute plagiarism and failure to comprehend and practice appropriate citation techniques" (Joy et al., 2013, p.4). A significant contributor to source code plagiarism is students' need for more understanding regarding academic integrity and citation methods. In some cases, students may be unaware of the consequences of copying code without proper attribution, viewing it as a permissible practice. Another major problem with source code plagiarism is the extent to which "acceptable collaboration" is allowed. "Confusion about the acceptable limits of collaboration is a particularly difficult problem in relation to source-code, since students are often encouraged to help each other learn by discussing their work" (Joy et al., 2013, p.15). The difference in collaboration between school and the workforce can create confusion for many students. "Collaborative system development in the software industry is common practice and reuse of source code within an organization would be encouraged to save both time and money. This approach is at odds with the individual approach necessary for assessment purposes" (Joy et al., 2013, p.16). The lack of education on what constitutes plagiarism and appropriate collaboration during assignments has led to the widespread source code plagiarism we see today.

4. Impact on Education and Individuals

How exactly does this affect our society? Source code plagiarism in education may not primarily affect the current generation of coders. However, it will have lasting effects on the future generation. The effects can be separated into two main categories: educational and mental. Educational effects stem from source code plagiarism's effects on the education system. "As the digital age is evolving, thereby increasing student access to information, it is becoming more difficult for academic institutions to maintain academic integrity across instructional programs" (Buitendag et al., 2013, p.46). Plagiarized assignments distort the evaluation process, making it challenging to gauge students' actual skills and comprehension of the subject matter. This, in turn, erodes the fairness of assessments and undermines the credibility of the educational institution's grading system. Students who fall to the temptation of source code plagiarism may be prone to carrying similar practices to other subjects, thus disheveling the education system.

The mental effects source code plagiarism has on students are also alarmingly detrimental. If this continues, many students incapable of writing their code may be brought into the workforce without the proper qualifications. Because they achieved their previous success in their school years, they may not be up to par when replicating that in the workforce. When students resort to copying code rather than working through programming challenges, they miss valuable opportunities to develop problem-solving skills and gain a deeper understanding of coding concepts. This hampers their growth as future programmers and engineers.

5. Proposed Solutions

How can we counteract source code plagiarism? The potential solutions to source code plagiarism can be separated into two categories: reactive and proactive methods. "Reactive" methods focus on catching plagiarism instead of preventing it (Buitendag et al., 2013, p.47). "Proactive" methods, on the other hand, focus on preventing plagiarism before it occurs (Buitendag et al., 2013, p.47).

Reactive methods use a combination of algorithms and techniques to create a detection engine suitable for the user. Multiple techniques have been used to find the best method to detect plagiarism successfully. "Some of them focus more on effectiveness factors (such as accuracy and the capability to detect complex modification) while the others focus on the efficiency (such as processing time)" (Karnalim et al., 2019, p.322). However, "[most automated source code plagiarism detection typically works in two consecutive phases: tokenization and comparison" (Karnalim et al., 2019, p.323). "Tokenization converts source code files to an intermediate representation prior to comparison" (Karnalim et al., 2019, p.323). As tokenization is the most common available representation, it has been adopted in numerous studies, some of which have been matured as tools, such as Jplag and YAP (Karnalim et al., 2019, p.323). "The comparison phase measures the similarity degree between two source code files based on their intermediate representation" (Karnalim et al., 2019, p.323). The comparison phase can then further be categorized into techniques: "namely structure-based, attribute-based, and hybrid" (Karnalim et al., 2019, p.323).

As previously stated, proactive methods focus on preventing source code plagiarism before it occurs. Academic institutions implement proactive methods by "educating students on plagiarism, creating clear anti-plagiarism policies across different academic programs, and adopting honor codes" (Buitendag et al., 2013, p.47). The implementation of honor codes fosters a sense of community among students, discouraging academic dishonesty due to the commitment to upholding trust. These codes complement anti-plagiarism policies, strengthening the deterrent against plagiarism by enhancing awareness of its potential consequences. Contrary to popular opinion, not every form of plagiarism is done with the malicious intent of taking someone's code. It is often seen that students cite the source of the code but do so incorrectly, thus committing plagiarism. Educational institutions can implement programs and workshops to educate students about academic integrity, plagiarism, and proper citation practices. By fostering a culture of integrity, institutions can reduce the allure of source code plagiarism to students.

6. Challenges in Implementing Solutions

Whether an academic institution decides to implement a proactive or reactive approach to plagiarism, there are downsides to each one. Proactive approaches focus on the long term and show little to no results in the short term. On the other hand, reactive methods focus on the short term but create issues in the long term.

Problems such as false positives, language barriers, and division between mentor and mentee often arise when implementing reactive methods. Regardless of the algorithm that one uses to detect source code plagiarism, there is always the possibility of false positives arising. False positives may arise for many reasons, but the most common is if there are standard design patterns in code. Since the most common plagiarism detection tools, such as Jplag, use tokenization to compare pairs of codes, it will flag instances in code that are commonly used. An example of this would be "if and else" statements. If the final project for a coding class asks for specific variables to be included or excluded, then it is likely that an "if and else" method will be created. Unfortunately, Jplag would flag all the codes as similar for having the same methods, even though it might be the easiest method to include and exclude variables.

Another issue would be the language barriers that some algorithms have. As there are over two hundred languages worldwide, having an algorithm be able to work in every single language is difficult. There are nuances in every language that should be taken into account when checking for plagiarism. This can be seen in the SuaCode Africa 2.0 experiment (Boateng & John, 2021, p.209). The algorithm for checking plagiarism in the SuaCode Africa 2.0 experiment solely applied to English and French (Boateng & John, 2021, p.209). Similarly, this problem arises in different programming languages. As SuaCode Africa 2.0 uses Processing-based code files, coding languages like Python can not be effectively detected for plagiarism (Boateng & John, 2021, p.211).

Although reactive methods provide instant results in discovering pieces of plagiarism, they have a negative impact in the long term. Researchers stress that as the reactive methods add little value to the academic process, it could lead to students focusing on ways to bypass the engine instead of learning the material (Buitendag et al., 2013, p.47). This could result from

the students taking it as a personal attack, leading to students seeing their instructor "as the enemy instead of the mentor" (Buitendag et al., 2013, p.47).

Proactive methods, on the other hand, have different problems. As they focus on changing the student body as a whole, proactive methods require a significantly longer time to implement. As proactive methods fail to detect instances of plagiarism, students that do not care for honor codes or a sense of community are not deterred from it. The same problem arises when educational institutions fail to create clear anti-plagiarism policies that highlight the consequences that might arise if caught.

Conclusion

Source code plagiarism is a growing problem within educational settings, fueled by academic pressures, time constraints, and a lack of awareness about academic integrity. To combat this issue, educational institutions must adopt a multifaceted approach that combines education, clear guidelines, and the use of plagiarism detection tools. However, the implications of implementing these solutions must also be considered. Ultimately, computer science and programming courses must prioritize the cultivation of integrity to prevent source code plagiarism and nurture ethical coders for the future.

References

Boateng, G., & John, S. (2021). “I didn’t copy his code”: Code plagiarism detection with visual proof. Lecture Notes in Computer Science, 208–212. https://doi.org/10.1007/978-3-030-78270-2_37

Boyatt, R., Cosma, G., Joy, M., Sinclair, J., & Yau, J. (2013). Student perspectives on source-code plagiarism. *International Journal for Educational Integrity*, 9(1). https://doi.org/10.21913/ijei.v9i1.844

Budi, S., Joy, M., Karnalim, O., & Toba, H. (2018, November 30). Source code plagiarism detection in academia with Information Retrieval: Dataset and the observation. Informatics in Education. https://eric.ed.gov/?id=EJ1233501

Buitendag, A. A. K., Hattingh, F., & van der Walt, J. S. (2013). Presenting an alternative source code plagiarism detection framework for improving the teaching and learning of programming. *Journal of Information Technology Education: Innovations in Practice*, 12, 045–058. https://doi.org/10.28945/1769

Cheers, H., Lin, Y., & Smith, S. P. (2021). Academic source code plagiarism detection by measuring program behavioral similarity. IEEE Access, 9, 50391–50412. https://doi.org/10.1109/access.2021.3069367

Clark, D., Krinke, J., & Ragkhitwetsagul, C. (2016). Similarity of source code in the presence of pervasive modifications. 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM). https://doi.org/10.1109/scam.2016.13

Gondaliya, T. P., Joshi, H., & Joshi, H. D. (2014). Source Code Plagiarism Detection ‘SCPDet’: A Review. *International Journal of Computer Applications*, 105(17).

Krutsch, E. (2022). Computer Science Education Week: Explore in-demand it jobs. U.S DOL Blog. https://blog.dol.gov/2022/12/01/computer-science-education-week-explore-in-demand-it-jobs