Mitigating plagiarism and collusion in K-12 as initial knowledge for programming instructors in higher education

Oscar Karnalim, Mewati Ayub & Hapnes Toba

Maranatha Christian University
Bandung, Indonesia

ABSTRACT: Programming plagiarism and collusion should be mitigated in higher education. For that purpose, instructors are expected to understand students' perspective about the matter, and how plagiarism and collusion were mitigated at earlier educational levels. Hence, a study was undertaken to investigate 42 K-12 teachers how they mitigated plagiarism and collusion in their classes. The collected data were summarised via the latent Dirichlet allocation (LDA) topic modelling technique. It was found that K-12 teachers only generally informed students about plagiarism and collusion. Penalties for the misconduct varied, but mostly there were no serious consequences. Students whose work was copied could be identified through their academic performance, but they were expected to be penalised as well. The teachers detected plagiarism and collusion based on student behaviour; however, they provided opportunity for students to explain the situation and prove innocence. It is recommended that programming plagiarism and collusion be comprehensively explained by instructors, stricter detection strategies be employed, and more severe penalties be imposed when justifiable.

INTRODUCTION

In programming, plagiarism and collusion are two common breaches of academic integrity in higher education [1]. They are essentially about the reuse of program code without proper acknowledgment [2]. The only difference is that exclusive to collusion, the original (yet unauthorised and unacknowledged) owner of the copied code is aware of the misconduct and allows that to happen. In some definitions, collusion is often considered as part of plagiarism [3]. Both plagiarism and collusion have been more concerning during the current pandemic, especially due to the lack of supportive environment [4][5].

To deal with plagiarism and collusion in programming, instructors need to reduce opportunity to cheat and/or mitigate the pressure [6]. They can also derationalise students' own justifications to engage in the misconduct. It is preferred to involve these three strategies at once, but if it is not practical, at least one of them should be employed.

Reducing opportunity to cheat means making cheating difficult. Instructors are encouraged not to reuse past assessments without any modifications [7]. It is also suggested to generate different versions of assessments [8] and/or personalise the assessments with students' own case studies [9]. Further confirmation of the authorship of a specific work can also be employed like, for example, an oral presentation [10].

Mitigating pressure to cheat means dealing with factors that can make students stress about the assessments. Instructors can incentivise early submissions to deal with time pressure [11]. They can also issue many small assessments instead of few but larger ones to deal with the pressure of assessment and its difficulty [12].

Derationalising student justifications means informing students about programming plagiarism and collusion, including instructors' expectation about the matter in their courses [13]. The information is typically delivered at the beginning of the course or right after a particular assessment is issued. It is also suggested to integrate such information as part of the curriculum [14] or to use educational tools focusing on the matter [15][16].

The information for students about plagiarism and collusion should be focused on what students might misinterpret. A number of studies have been conducted to capture students' perspective about plagiarism and collusion. Students are found to have misconceptions about self-plagiarism (i.e. reusing their own code without acknowledgment) [17], reusing source code from a textbook [18], copying an early draft of a colleague's work [17], asking someone to fix code errors [17] and about collusion in general [18].

To mitigate programming plagiarism and collusion in higher education, it is important to understand not only students' prior knowledge about the matter, but also how plagiarism and collusion are mitigated in K-12 education. The authors

believe that by knowing how students were earlier advised about plagiarism and collusion in K-12, higher education instructors can be more effective in educating them about programming plagiarism and collusion at a later stage. The instructors can confirm whether the aspects that students misunderstand in higher education were also misunderstood in K-12. If so, more efforts are necessary to educate students about these aspects. The instructors can also prepare better explanation for those aspects that students understood in K-12, but do not understand in higher education.

Consequently, this study was focused on how K-12 teachers deal with such a matter in their classes using a questionnaire survey and the latent Dirichlet allocation (LDA) topic modelling technique [19][20]. In this article, a summary of the study is outlined, together with key recommendations for programming instructors in higher education. To the best of the authors' knowledge, and in this context, this is one of the first such studies (i.e. capturing how plagiarism and collusion are dealt with in K-12 and mapping the information for higher education instructors to maintain academic integrity in programming).

The study was conducted in the country of residence of the first author with 42 K-12 teachers (19 primary and 23 secondary). The survey was distributed at the end of a hybrid workshop for K-12 teachers about Bebras challenge, a computational thinking competition for K-12 students [21].

METHOD

How plagiarism and collusion are mitigated in K-12 education was captured via a questionnaire survey, comprising seven questions. Six of them are open-ended, while one (Q5) is close-ended with binary responses (*yes* or *no*). The questions can be seen in Table 1.

Table 1: Survey questions.

ID	Question
Q1	How do you inform students about plagiarism and collusion?
Q2	When do you inform students about plagiarism and collusion?
Q3	What are the penalties for students involved in plagiarism and collusion?
Q4	How do you detect plagiarism and collusion?
Q5	Do you allow students suspected of plagiarism and collusion to prove their innocence?
Q6	How do you differentiate students who copy the work from students whose work is copied?
Q7	What are the penalties for students whose work is copied?

Q1 asks about how students are informed about plagiarism and collusion. Programming instructors typically explain what constitutes plagiarism and collusion in various level of detail.

Q2 asks about the timing for conveying this information to students. In programming education, the information is often delivered in early weeks and/or after an assessment is issued.

Q3 asks about the penalties for those involved in plagiarism and collusion. Students need to be penalised to make them aware that such dishonest acts are discouraged. Programming instructors give zero marks on the corresponding assessment or the final grade of the course.

Q4 asks about how plagiarism and collusion are detected. In programming, some instructors do the detection manually, while others take advantage of automated similarity detectors like MOSS [22] (an example of a common tool) and CSTRANGE [23] (an example of a recent tool).

Q5 asks about the opportunity for students suspected of plagiarism and collusion to prove their innocence. Some programming instructors provide this opportunity for non-obvious acts of copying.

Q6 asks about how K-12 educators differentiate students who copy the work from those whose work is copied. Sometimes, the former are penalised with higher consequences than the latter.

Q7 asks about the penalties for students whose work is copied, and whether they are comparable with those of students who copy the work.

The survey was distributed at the end of a hybrid workshop for K-12 educators at the first author's institution. The workshop was about a computational thinking competition called Bebras challenge [21]. Participation was voluntary and the responses were collected via Google Forms.

For all the questions except Q5, a bigram-based LDA topic modelling approach [19][20] was employed. A bigram model, i.e. a sequence of two adjacent elements from a string of words is expected to report relevant words with stronger semantic relations in the textual description [24][25]. The national language of the first author, in particular, has numerous phrases with repetitive words that are meaningless if separated.

To tune the optimal number of topics in each question, a grid search for k = 2 to 10 topics was run, and the highest topic's coherence score was chosen to be considered the best. A topic relation graph was constructed to investigate how one word was related to the other words in a topic and how they were related to other topics. The topic relation graph would also be useful to see the topic centrality and could be considered a core keyword for constructing hypothetical concepts to the open-ended questions.

The construction of a topic relation graph has two goals. First, to relate the highest-weighted word in each topic to all the other words in the same topic. Second, to relate some words in a topic to some other words in another topic, i.e. the words that occur in multiple topics. In this way, it would be possible to follow a path from one core keyword to other words in other topics to create hypothetical sentences or phrases. The topic relation graph would also enable to identify whether exclusive topics exist in the generated topic set.

Exclusive to Q5, the proportion of each response (*yes* or *no*) was recorded and discussed. Based on the findings, a list of recommendations is provided for higher education instructors to inform students about programming plagiarism and collusion.

It is worth noting that in Indonesia, K-12 students currently do not have mandatory programming courses. However, given that the most substantial difference between general plagiarism and collusion with those in programming is the media (i.e. text versus code), the authors of this article opine that knowledge about general plagiarism and collusion in K-12 education can be advantageous in managing and mitigating programming plagiarism and collusion in higher education.

The authors are also aware that how plagiarism and collusion are handled at the primary level are different to that at the secondary level: the former tends to be more lenient than the latter. However, it is believed that the educators hold comparable values in maintaining academic integrity.

RESULTS

To identify the best number of topics, the coherence scores were sorted in each iteration and the highest score was selected. The results of the coherence score for each question can be seen in Table 2. The scores were evaluated and it was found that the score differences are not statistically significant (p = 0.05).

Table 2 shows that some questions have various number of topics. More topics would indicate more variations in the conceptual decision during sentence construction to answer a question. In reality, this suggests that a lower number of topics means more limited responses.

Table 2: Coherence scores in each k iteration; the shaded cells indicate the best number of topics.

k	Q1	Q2	Q3	Q4	Q6	Q7
2	0.529	0.674	0.583	0.648	0.618	0.511
3	0.513	0.675	0.538	0.643	0.663	0.506
4	0.547	0.678	0.561	0.589	0.625	0.534
5	0.572	0.678	0.597	0.578	0.605	0.548
6	0.495	0.689	0.554	0.578	0.574	0.617
7	0.500	0.680	0.583	0.585	0.617	0.608
8	0.489	0.672	0.570	0.561	0.619	0.593
9	0.493	0.674	0.543	0.585	0.646	0.575
10	0.491	0.674	0.578	0.563	0.612	0.582

For instance, to address Q3 (penalties for students involved in plagiarism and collusion), there are five topics with some alternatives of …*how to punish a cheating student*. These are such as: inviting students' parents for reflection, verbal warning, marking the assessments with zero, reducing some marks, and retaking examinations or assessments with a different set of questions.

To address Q4 (how to detect plagiarism and collusion), on the contrary, there are only two topics. This might indicate stronger agreement among teachers: plagiarism and collusion are generally detected based on direct observation of student behaviour. Cheating students are often nervously looking at their surroundings instead of focusing on the examination.

All the questions except Q5 form one topic relation graph each. However, only one of them is shown here for brevity. Figure 1 shows the topic relation graph for Q4: how to detect plagiarism and collusion. There are two topics and each of those has a number of related words connected to one another. Topic centrality, i.e. the word which has the highest weight in a particular topic, is given in shaded circles.

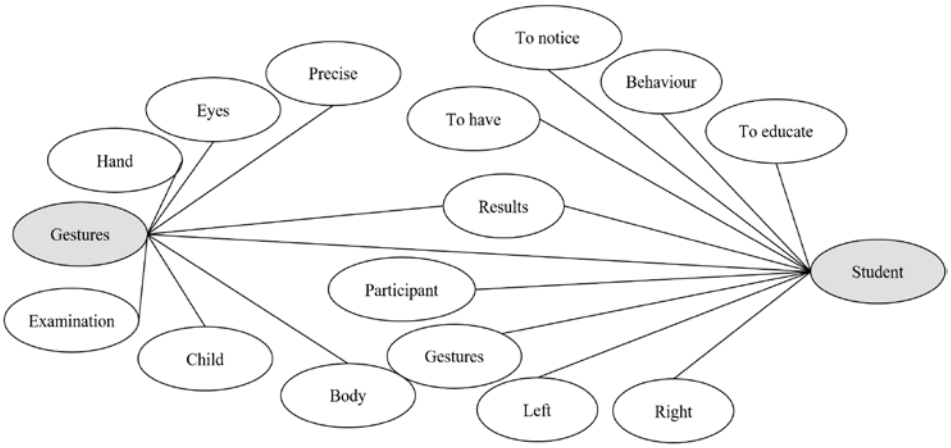

Figure 1: Topic relation graph for Q4.

MITIGATING PLAGIARISM AND COLLUSION IN K-12

In regard to Q1, teachers generally refer to plagiarism and collusion in a single term: cheating. Students are verbally informed that cheating means imitating and/or stealing other people's work. Further, they are explicitly prohibited to do such a thing as it is dishonest and harmful. A response states: *Cheating is dishonest and harmful to obtain a high score*. Some teachers state that cheating means …*lying to ourselve*s and it is …*a sign of limited self-confidence*.

In regard to Q2, teachers generally inform students about plagiarism and collusion while they are teaching. Sometimes, they remind them before tests or examinations, at morning reflection, or while reviewing tests or examinations. A response states: [Students are] *generally informed during the teaching session and right before the examination*.

Responses for Q3, regarding penalties for students involved in plagiarism and collusion include inviting students' parents for reflection, verbal warning, marking the assessments with zero, reducing some marks, and retaking examinations or assessments with a different set of questions. These are partly reflected in this response: *The student will get zero marks and their parents will be invited for reflection*. If misconduct is identified during examinations, students are immediately expelled from the examination room. A teacher states: [Students] *will lose some marks with their answer sheet collected immediately. They will be forced to leave the examination room.*

Responses to Q4 indicate that plagiarism and collusion are generally detected based on student behaviour, especially during examinations. A response states: *Student's gesture and attitude are useful for the detection*. Another response states: *Students look to the right and the left side instead of focusing on the examination can be suspicious*.

In regard to Q5, most teachers (86%) state that they allow students suspected of plagiarism and collusion to prove their innocence. This is a good thing as not all suspicions are justifiable.

Responses to Q6 indicate that teachers can generally differentiate students who copy the work from those ones whose work is copied based on their academic performance. Students whose work is copied are typically smarter than their counterparts. Academic performance can be identified from previous tests, examinations, and/or assessments. Sometimes, it can also be seen from how students respond to the teaching materials. A teacher responds …[they are] *differentiated based on daily student activities and former test scores*. Another adds: *Students who copy the work cannot explain the answers precisely in detail*.

Responses to Q7 show that students whose work is copied are usually penalised in the same manner as those who copy the work. Example penalties are inviting students' parents for reflection, verbal warning, marking the assessments with zero and reducing some marks. In regard to collusion in the academic environment it is more likely that students whose work is copied are aware about the misconduct. A teacher states: *If a student provides the answers* [to others] *intentionally, their marks will be still zero*. Another response states: *Students need to repeat the examination if they are proved to give their answers* [to other students]. However, some teachers suspect plagiarism rather than collusion, and they state that …*if the answers are proved to be stolen, the student who originally owns the answer will only be advised to be more careful*.

RECOMMENDATIONS FOR PROGRAMMING INSTRUCTORS IN HIGHER EDUCATION

K-12 teachers only generally inform students about plagiarism and collusion (Q1). Hence, it is important for programming instructors to elaborate the matter more comprehensively, especially point out the differences between conventional plagiarism and collusion highlighting that in programming [26]. It is also suggested to focus on common misconceptions: self-plagiarism [17], reusing code from textbook [18], copying an early draft of a colleague's work [17] and asking someone to fix code errors [17] as indicated earlier in this article.

Both plagiarism and collusion are referred to as cheating by K-12 teachers. There is a need to advise students what constitutes plagiarism and what is collusion [18]. All of such information should be written in a document accessible by students at any time. If such information is only provided verbally like in K-12 education, students might forget about it at later points of the study period.

K-12 teachers educate students about plagiarism and collusion (Q2) in a similar manner as programming instructors: during the lecture, before tests or examinations or while reviewing tests or examinations. In addition to those periods of time, programming instructors are expected to inform students about the matter after issuing assessments. It is also suggested to show the importance of maintaining academic integrity while completing assessments. The collected responses of K-12 teachers show that such information seems to be overly focused on tests or examinations. Reminding students during morning reflection is not applicable for programming instructors as higher education rarely has such a reflection session.

K-12 teachers impose various penalties for students involved in plagiarism and collusion (Q3). However, most of them have insignificant consequences: inviting students' parents for reflection, verbal warning, reducing some marks, and retaking examinations or assessments with different set of questions. To show the importance of maintaining academic integrity, it is recommended for programming instructors to focus on severe penalties, such as marking the assessments with zero and/or failing the students from the course.

K-12 teachers detect plagiarism and collusion based on their observation of student behaviour (Q4). While the mechanism is effective and provides strong evidence, it is only applicable on onsite tests, examinations or assessments. Programming courses tend to have more take home tests, examinations or assessments. It is recommended for programming instructors to check the submissions for similarity, preferably with automated similarity detectors like MOSS [22] or CSTRANGE [23].

K-12 teachers provide the opportunity for students suspected of plagiarism and collusion to prove their innocence (Q5). Some programming instructors have already done the same, especially if the evidence is moderate and/or the marks are crucial to pass the course. Students can be asked to meet the instructors privately and provide their arguments.

K-12 teachers differentiate students who copy the work from students whose work is copied based on their academic performance (Q6). This is again, applicable for programming courses, especially those issuing several small assessments [12]. Students whose work is copied tend to have higher marks as they are more committed, motivated or knowledgeable.

K-12 teachers impose comparable penalties for students whose work is copied as for those students who copy the work. It is a good thing given that collusion tends to be more common in academia. Further, it is difficult to prove student intention of sharing their work. Programming instructors have already done the same.

To sum up, programming instructors are expected to provide detailed information about plagiarism and collusion, especially in the concept of programming and common misconceptions. They are also expected to show the importance of maintaining academic integrity by informing students about the matter more frequently, putting more severe penalties and employing automated similarity detectors for checking similarities across submissions.

It is important to provide the opportunity for students suspected of plagiarism and collusion to prove their innocence. However, comparable penalties should be imposed for all students: the ones whose work is copied and those who copy the work. Although students whose work is copied can be differentiated based on their academic performance, there is a fair chance that they have already been aware about the misconduct and let that happen willingly.

This study demonstrates that the aspects misunderstood by students in higher education, were also misunderstood in K-12 education; however, there are no aspects that would be misunderstood in higher education, if they were understood in K-12 education, which points out to the importance of early guidance in regard to plagiarism and collusion. Also, based on current practice, instructors are expected to allocate more time in educating students on common misconceptions.

CONCLUSIONS AND FUTURE WORK

This article reports on how plagiarism and collusion are mitigated in K-12 education via a questionnaire survey, and how to improve the current approach of dealing with programming plagiarism and collusion in higher education. Programming instructors are expected to comprehensively explain programming plagiarism and collusion, and especially point out the differences between the conventional form of such misbehaviour and common misconceptions. They are also expected to impose more severe penalties and stricter detection strategies. Although students can be provided with the opportunity to prove their innocence, those whose work is copied should still be penalised if they are involved in collusion.

For future work, there is a plan to replicate this study with more respondents, and if possible, respondents from other countries. It would also be useful to employ different research methods for richer findings, such as grouping teachers based on their students' age range. The plan would also include capturing high-school students' perspective about plagiarism and collusion as additional knowledge, given that they are somewhat fluent in information and communication technologies [27].

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support provided by Maranatha Christian University.

REFERENCES

1. Simon, Cook, B., Sheard, J., Carbone, A. and Johnson, C., Academic integrity: differences between computing assessments and essays. *Inter. Conf. on Computing Educ. Research*, 23-32 (2013).
2. Fraser, R., Collaboration, collusion and plagiarism in computer science coursework. *Informatics in Educ.*, 13, **2**, 179-195 (2014).
3. Cosma, G. and Joy, M., Towards a definition of source-code plagiarism. *IEEE Trans. on Educ.*, 51, **2**, 195-200 (2008).
4. Safri, N.M. and Sheikh, U.U., Issues and challenges of technology-enhanced learning during the Covid-19 era: a case study. *World Trans. on Engng. and Technol. Educ.*, 20, **2**, 89-94 (2022).
5. Rauzana, A. and Dharma, W., The effectiveness of on-line learning at an Indonesian university during the Covid-19 pandemic: students' perspectives. *World Trans. on Engng. and Technol. Educ.*, 20, **1**, 71-75 (2022).
6. Albluwi, I., Plagiarism in programming assessments: a systematic review. *ACM Trans. on Computing Educ.*, 20, **1**, (2019).
7. Simon, Designing programming assignments to reduce the likelihood of cheating. *Australasian Computing Educ. Conf.*, 42-47 (2017).
8. Fowler, M. and Zilles, C., Superficial code-guise: investigating the impact of surface feature changes on students' programming question scores. *ACM Technical Symp. on Computer Science Educ.*, 3-9 (2021).
9. Bradley, S., Creative assessment in programming: diversity and divergence. *Fourth Conf. on Computing Educ. Practice*, 13:1-13:4 (2020).
10. Halak, B. and El-Hajjar, M., Plagiarism detection and prevention techniques in engineering education. *The 11th European Workshop on Microelectronics Educ.*, 1-3 (2016).
11. Spacco, J., Fossati, D., Stamper, J. and Rivers, K., Towards improving programming habits to create better computer science course outcomes. *ACM Conf. on Innov. and Technol. in Computer Science Educ.*, 243-248 (2013).
12. Allen, J.M., Vahid, F., Downey, K. and Edgcomb, A.D., Weekly programs in a CS1 class: experiences with auto-graded many-small programs (MSP). *ASEE Annual Conf. & Expo.*, 1-13 (2018).
13. Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A. and Sinclair, J., Informing students about academic integrity in programming. *The 20th Australasian Computing Educ. Conf.*, 113-122 (2018).
14. Greening, T., Kay, J. and Kummerfeld, B., Integrating ethical content into computing curricula. *Sixth Australasian Conf. on Computing Edu.*, 91-99 (2004).
15. Karnalim, O., Simon, Chivers, W. and Panca, B.S., Educating students about programming plagiarism and collusion via formative feedback. *ACM Trans. on Computing Educ.*, 22, **3** (2022).
16. Le, T., Carbone, A., Sheard, J., Schuhmacher, M., De Raath, M. and Johnson, C., Educating computer programming students about plagiarism through use of a code similarity detection tool. *Inter. Conf. on Learning and Teaching in Computing and Engng.*, 98-105 (2013).
17. Karnalim, O., Kautsar, I.A., Aditya, B.R., Udjaja, Y., Nendya, M.B. and Darma Kotama, I.N., Programming plagiarism and collusion: student perceptions and mitigating strategies in Indonesia. *IEEE Inter. Conf. on Engng., Technol. & Educ.*, 9-14 (2021).
18. Zhang, D., Joy, M., Cosma, G., Boyatt, R., Sinclair, J. and Yau, J., Source-code plagiarism in universities: a comparative study of student perspectives in China and the UK. *Assess. & Evaluation in Higher Educ.*, 39, **6**, 743-758 (2014).
19. Blei, D.M., Ng, A.Y. and Jordan, M.I., Latent Dirichlet allocation. *J. of Machine Learning Research*, 3, **1**, 993-1022 (2003).
20. Chauhan, U. and Shah, A., Topic modeling using latent Dirichlet allocation: a survey. *ACM Computing Surveys*, 54, **7**, 1-35 (2021).
21. Dagiene, V. and Stupuriene, G., Bebras - a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Educ.*, 15, **1**, 25-44 (2016).
22. Schleimer, S., Wilkerson, D.S. and Aiken, A., Winnowing: local algorithms for document fingerprinting. *ACM Inter. Conf. on Manage. of Data*, 76-85 (2003).
23. Karnalim, O., Simon and Chivers, W., Layered similarity detection for programming plagiarism and collusion on weekly assessments. *Computer Applications in Engng. Educ.*, 30, **6**, 1739-1752 (2022).
24. Garg, M., UBIS: Unigram bigram importance score for feature selection from short text. *Expert Systems with Applications*, 195, 116563 (2022).
25. Ardito, G., A close reading and analysis of the New York state computer science learning standards. *Inter. J. on Integrating Technol. in Educ.* (2022).
26. Simon, Cook, B., Sheard, J., Carbone, A. and Johnson, C., Academic integrity perceptions regarding computing assessments and essays. *The 10th Annual Conf. on Inter. Computing Educ. Research*, 107-114 (2014).
27. Senjaya, W.F., Karnalim, O., Handoyo, E.D., Santoso, S., Tan, R., Wijanto, M.C. and Edi, D., Information and communication technology awareness of Indonesian high school students. *Global J. of Engng. Educ.*, 20, **3**, 217-223 (2018).