Source Code Plagiarism Detection in an Educational Context: A Literature Mapping

Rodrigo C Aniceto
Universidade de Brasília
Brasília, Brazil
rodrigoaniceto@ymail.com

Maristela Holanda
Universidade de Brasília
Brasília, Brazil
0000-0002-0883-2579

Carla Castanho
Universidade de Brasília
Brasília, Brazil
0000-0002-7328-5479

Dilma Da Silva
Texas A&M University
College Station, US
0000-0001-6538-2888

***Abstract*—Detection of plagiarism in students' source codes in college-level programming courses is an important topic for instructors and institutions that seek to pursue project-based learning while enforcing honor codes and maintaining traditional grade-based skill assessment methods. There are different approaches for plagiarism detection currently being researched. This paper aims to answer the question: What does the literature report on source code plagiarism detection in university settings? To answer that, we used a systematic mapping process of recent literature. We selected 109 papers published between 2015 and 2020 that deal with this subject specifically in an educational context. We found that this research area is currently expanding and being studied worldwide. There were papers from 37 different countries, and the number of publications per year has been increasing since 2017. The most targeted programming languages are Java, C++, C, and Python. The most studied plagiarism detection tools are MOSS, JPlag, SIM, Plaggie, and Sherlock. Our study also identified new methodologies created to tackle this problem, such as the analysis of students' typing patterns or their coding style. We noticed that the proposed solutions are mainly based on static source code analysis instead of following the development process. This paper describes our findings.**

***Index Terms*—programming, plagiarism detection, source code, academia, education, similarity, mapping**

I. Introduction

The large amount of information available online has made it easier to commit plagiarism in the academic environment. Plagiarism can be defined as when someone presents another person's ideas or work as their own [1]. This type of problem cannot be ignored primarily for ethical reasons. Plagiarism can also damage the reputation of institutions and the development of students who do so.

A specific type of plagiarism to be studied is the plagiarism of source codes in programming assignments at universities. Detecting students who copy the source code from others is an area of investigation with considerable relevance for institutions seeking to improve the quality of their teaching, as well as to increase the reputation of the professionals they are educating. This research area has produced a large number of papers describing experiences with detection tools already used in the academia [2] as well as suggestions for new solutions to deal with plagiarism [3] [4] [5].

In this context, this paper aims to answer the research question: What does the literature report on source code plagiarism detection in university settings? We divided this question into other six research questions and used a systematic mapping process of recent literature to answer them.

A search was made in the Scopus and Web of Science academic databases, followed by the application of inclusion and exclusion criteria, resulting in a set of 109 papers that were analyzed in order to answer the questions.

The rest of this paper is organized as follows. In Section II the methodology and the research questions are described. Section III details the papers selection process and in Section IV we answer the research questions using the data obtained. In Section V we discuss the results, and the conclusions and future work are presented in Section VI.

II. Research Questions

In this work we employed the systematic literature mapping process based on [6], composed of four stages: (i) define the research questions; (ii) select the relevant papers; (iii) analyze the papers; and (iv) answer the research questions.

Our main research question is (RQ): What does the literature report on source code plagiarism detection in university settings? To answer this question, we divided it into six other research questions, listed below:

- RQ1: What is the number of papers published per year?
- RQ2: In which countries were these papers produced?
- RQ3: Which conferences and journals have published most papers on this subject?
- RQ4: What data sources are used to detect plagiarism?
- RQ5: Which programming languages are most used in plagiarism detection experiments?
- RQ6: What are the main plagiarism detection tools used in the last few years?

The next steps of the mapping process are described in the following sections.

III. Selecting the relevant papers

The stage of selecting the relevant papers is fundamental in any literature mapping process, which involves defining a search string, choosing the academic databases in which to search for the papers, and the inclusion/exclusion criteria to decide the significance of each publication in the study.

To define the search string to be used in the search, there must be keywords related to the subject of this investigation, but they cannot be too broad or there will be many false

positives. The first version was:

"source code" AND ("fraud detection" OR "plagiarism")

This version had a low number of results (335 on Scopus on July 2021). We also noted that the term "fraud detection" is widely used for financial fraud, but less used in educational environments. On the other hand, the term "cheating" is quite common and has been added, also many papers use only "code" instead of "source code".

Several query string tests were done after this one. We also used the VOSviewer tool [7] to help analyze the keywords and obtained expert opinions on keywords to add. In the end, the following query string was employed:

("code" OR "algorithm" OR "algorithms" OR "programming" OR "Computer Science Education") AND ("plagiarism" OR "similarity detection" OR "cheat" OR "cheating")

The academic databases selected in this study are Scopus and Web of Science because they index the most important conferences and journals related to Computer Education, such as IEEE FIE, ACM SIGCSE, ACM and IEEE journals.

In the inclusion/exclusion criteria stage, we first defined inclusion criteria, while the exclusion criteria were applied in two cycles detailed in the subsections below. The inclusion criteria are:

- IC1: Papers must be written in English.
- IC2: Papers must be published between 2015 and 2020.
- IC3: Papers must be from the "Engineering" or "Computer Science" areas.
- IC4: Papers must be of the type "conference paper", "journal article" or "book chapter".

Figure 1 shows the process to select the relevant papers and the research cycles. The search string with the inclusion criteria was applied on July 1st, 2021, and returned 901 papers from Scopus and 692 from Web of Science. From a total of 1593, we removed 317 duplicate papers, thus resulting in 1276 selected papers, as shown in Table I.

TABLE I
SEARCH RESULTS

Source	Number of Papers
Scopus	901
Web of Science	692
Duplicated results	317
Final Results	**1276**

A. First cycle

The 1276 filtered papers were used in the first cycle of the research. We collected their metadata and put them in a spreadsheet. The process consists of eliminating all papers that are clearly unrelated to our study based on the metadata.

The metadata was: title, authors, source, abstract, year, publisher, document type, source title, volume, issue, cited by, DOI, link, publication stage, EID, country and page count.

To remove the unrelated papers, we applied the first exclusion criteria:

- EC1: Remove all papers that are not related to plagiarism detection in source code.

We removed several papers related to text plagiarism. Also, there were many works on image/audio processing, computer networks, malware detection, video game cheating, biology, etc, which were withdrawn. At the end of this cycle, 204 papers remained in our spreadsheet.

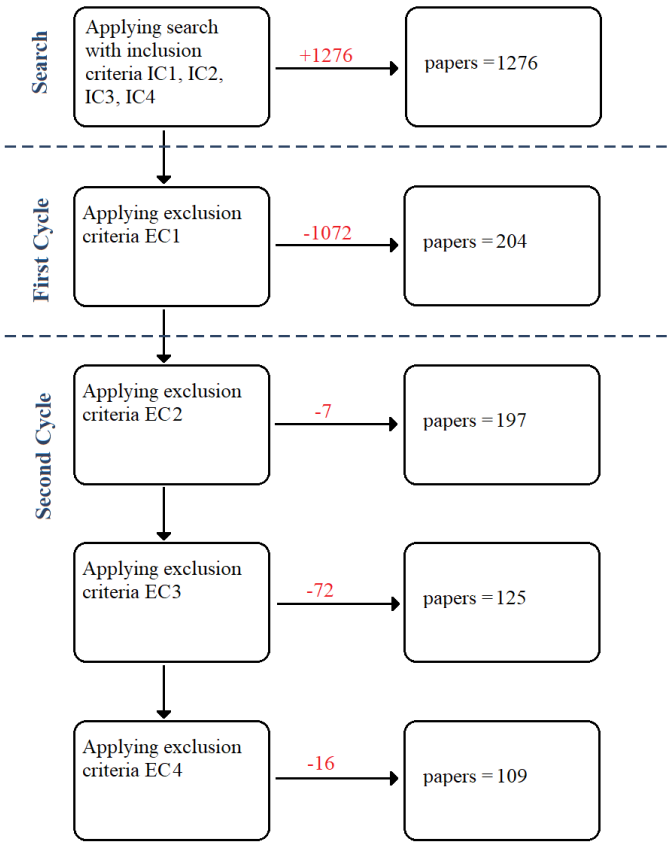

Fig. 1. Summary of inclusion/exclusion criteria in the research stages.

B. Second cycle

The second cycle consisted of removing the papers related to the detection of plagiarism, but with little or no relevance for our literature mapping. The following exclusion criteria were applied:

- EC2: Remove all papers with 3 pages or less.
- EC3: Remove all papers that are not related to college-level education.
- EC4: Remove all papers that do not deal with technology.

Initially, we removed 7 papers that were three pages or less because they are classified as short papers and tend to have less details in their research.

Then, 72 papers were removed because they were not related to university-level education. They were mostly about reusing source codes in scientific or business contexts.

After that, we found that there were some papers that referred to cheating in programming assignments, but had an emphasis on social sciences or were not related to technology. We removed 16 papers with such profile, resulting in 109 papers in our spreadsheet at the end of this cycle.

IV. Results and Analysis

In this session we will answer the six research questions based on the set of 109 selected papers.

A. RQ1: What is the number of papers published per year?

The number of papers published per year is shown in Figure 2. There was a drop in this number in 2017, followed by an increase in subsequent years. It is noticeable that the number of publications raised considerably in 2019 and 2020.

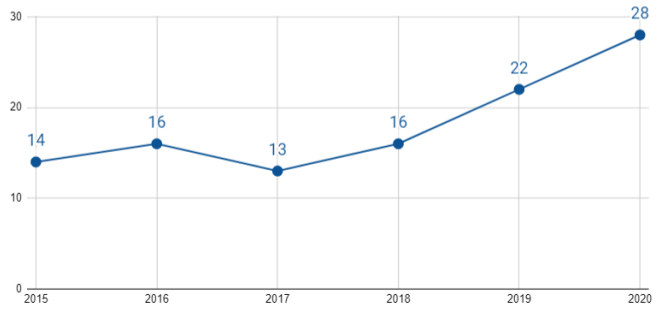

Fig. 2. Number of papers per year.

B. RQ2: In which countries were these papers produced?

We identified publications from 37 countries in 6 different continents. Figure 3 shows a map indicating the distribution of these countries.

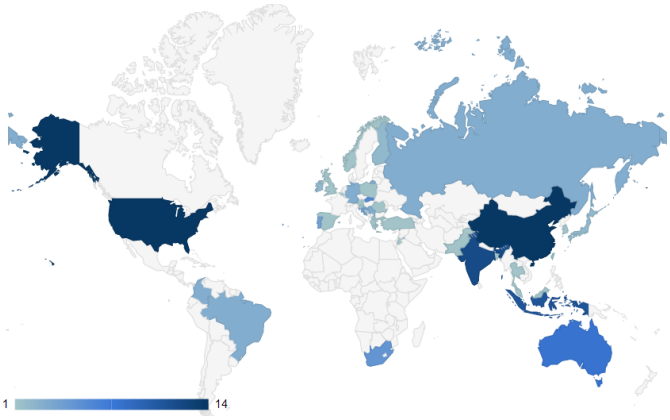

Fig. 3. Distribution of the papers by countries.

The 19 countries that produced the largest number of papers are shown sorted in Table II. The countries with the most publications were China and United States, followed by India and Indonesia. Some papers appear repeated in the table because they are from more than one country.

TABLE II
COUNTRIES WITH THE MOST PUBLICATIONS

Country	No.	Papers
China	14	[8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21]
United States	14	[22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35]
India	12	[36] [37] [38] [39] [40] [41] [42] [43] [5] [44] [45] [46]
Indonesia	11	[47] [48] [49] [50] [51] [52] [53] [4] [54] [55] [56]
Australia	8	[57] [58] [59] [60] [61] [4] [55] [56]
Slovakia	6	[62] [63] [3] [64] [65] [66]
South Africa	5	[67] [68] [69] [70] [71]
Croatia	4	[72] [73] [2] [74]
Portugal	4	[75] [76] [77] [78]
Brazil	3	[79] [80] [81]
Germany	3	[82] [25] [83]
Russian Federation	3	[84] [85] [86]
Serbia	3	[87] [88] [89]
United Kingdom	3	[2] [53] [90]
Colombia	2	[91] [65]
Finland	2	[19] [92]
Ireland	2	[93] [94]
Japan	2	[95] [96]
Taiwan	2	[97] [98]

In addition to the countries shown in Table II, we also found one publication from each of the following countries: Austria [84], Bangladesh [99], Belgium [100], Bosnia and Herzegovina [101], Greece [102], Jordan [103], Malaysia [104], North Macedonia [105], Norway [106], Pakistan [18], Poland [65], Romania [107], Spain [108], Slovenia [109], South Korea [110], Spain [111], Thailand [112] and Turkey [108].

C. RQ3: Which conferences and journals have published most papers on this subject?

The papers analyzed here came from 25 different journals and 66 conferences. Table III shows the five journals with the largest number publications and Table IV shows eleven conferences with two or more papers.

TABLE III
JOURNALS WITH THE MOST PUBLICATIONS

Journal	No.	Papers
Computer Applications in Engineering Education	4	[91] [50] [111] [108]
IEEE Access	2	[101] [66]
Informatics in Education	2	[72] [53]
Journal of Theoretical and Applied Information Technology	2	[103] [40]
Scientific Programming	2	[10] [17]

It is worth mentioning that there was a lot of diversity in the origin of these articles. Most of them came from different conferences or journals, with few repetitions between them.

TABLE IV
CONFERENCES WITH THE MOST PUBLICATIONS

Conference	No.	Papers
ICSESS - International Conference on Software Engineering and Service Sciences	4	[63] [58] [61] [48]
SIGCSE - ACM Technical Symposium on Computer Science Education	4	[29] [30] [31] [32]
ACE - Australasian Computing Education Conference	3	[57] [60] [4]
CSEDU - International Conference on Computer Supported Education	3	[8] [82] [106]
FIE - IEEE Frontiers in Education Conference	3	[13] [81] [20]
Koli Calling - International Conference on Computing Education Research	3	[55] [56] [92]
L@S - ACM Conference on Learning at Scale	3	[24] [26] [34]
MIPRO - International Convention on Information and Communication Technology, Electronics and Microelectronics	3	[73] [109] [74]
ICCSE - International Conference on Computer Science and Education	2	[19] [93]
SIIE - International Symposium on Computers in Education	2	[77] [78]
TURC - ACM Turing Celebration Conference	2	[11] [12]

D. RQ4: What data sources are used to detect plagiarism?

The second cycle resulted in 109 works, of which 93 specifically suggest new implementations of plagiarism detection software in programming assignments. These papers propose different solutions, but we could see that most of them are based only on the static analysis of the source code, that is, the analysis is done only once after the students finished their assignments.

On the other hand, we found some papers that propose alternative solutions regarding the source of the data to be used by the plagiarism detection tool:

- [22] and [77] follow the progress of the code while it is being developed. In this model, students make multiple commits during their development process and the software analyzes the differences between each submission aiming an indication of plagiarism.
- [101] proposed a solution similar to the above, but here students must use an online IDE that is constantly sending change reports. The tool does not need to make comparisons between the students' source codes. It follows the entire development process as if the tool were filming it. Its concept is based on the idea that the typing patterns of a student who is copying a work is different from the usual development process.
- [95], [51] and [86] use coding style in addition to code comparison. It is also possible to compare a student's coding style across different tasks, to check for any significant change.
- [16] proposes to investigate the behavior of students whose codes have the same level of similarity in different tasks. This may imply that they are committing plagiarism but hiding it well, so that the tools fail to detect it. This investigation is done in conjunction with an analysis of the coding style.
- [10] analyzes program execution logs in addition to code comparison. It uses these execution logs to create the flow charts of the programs. The similarity of these flow charts is then measured. This is done to detect plagiarism even when the student uses more complex obfuscation techniques.

The analysis of all these papers also showed that no method is 100% effective, and it is up to the teacher to decide on the suspected cases. An example of an action that the teacher can take in such cases is to ask suspicious students to explain their code orally to see if they really understood what they did [101].

E. RQ5: Which programming languages are most used in plagiarism detection experiments?

It is important to note that several of the papers analyzed here have implementations that are independent of the programming language or designed to work with many of them [52] [86] [80] [25] [64] [23] [35]. However, there are some programming languages that have received more attention from research. This may indicate that they are used more frequently in programming courses. As a result, they have more detection tools customized for them.

The programming languages used that we have identified are shown in Table V.

TABLE V
MOST USED LANGUAGES

Language	No.	Papers
Java	30	[111] [18] [59] [58] [61] [4] [10] [92] [43] [37] [47] [49] [48] [53] [106] [62] [109] [90] [5] [12] [107] [72] [103] [24] [111] [45] [66] [26] [55] [29]
C++	19	[57] [11] [91] [102] [50] [105] [85] [71] [69] [29] [22] [111] [18] [103] [89] [5] [12] [107] [19]
C	18	[99] [101] [79] [14] [17] [16] [13] [39] [44] [95] [108] [89] [103] [5] [12] [107] [20] [81]
Python	8	[111] [5] [103] [24] [9] [26] [55] [29]
C#	6	[63] [100] [3] [72] [103] [66]
PHP	5	[107] [77] [5] [107] [72]
JavaScript	2	[107] [72]
Go	1	[12]
Perl	1	[5]
Assembly	1	[89]

F. RQ6: What are the main plagiarism detection tools used in the last few years?

To answer this question, we first selected those papers that do comparative analysis or case studies of current plagiarism detection tools. The papers dedicated exclusively to doing that are [87] [57] [38] [69] [89] [73] [79] [2] [71].

Table VI gives some details on the 5 most used tools which are MOSS, JPLAG, SIM, Plaggie and Sherlock. Table VII shows a complete list of tools or technologies that were employed in the works we have analyzed.

TABLE VI
DETAILS OF THE MOST USED PLAGIARISM DETECTION TOOLS

Tools	Programming languages	Algorithm	GUI	Open source	URL
MOSS	C, C++, C#, Java, Python, etc	Winnowing	YES	NO	https://theory.stanford.edu/∼aiken/moss/
JPLAG	C, C++, C#, Java, Scheme	RKR-GST	YES	YES	https://jplag.ipd.kit.edu
SIM	C, Java, Pascal, Lisp, etc	Tokenization	NO	YES	https://dickgrune.com/Programs/similarity_tester
Plaggie	Java	(not available)	YES	YES	https://www.cs.hut.fi/Software/Plaggie
Sherlock	C++, Java	Tokenization	YES	YES	http://warwick.ac.uk/iasgroup/software/sherlock

TABLE VII
PLAGIARISM DETECTIN TOOLS LIST

Tools	**No.**	**Papers**
MOSS	9	[87] [57] [38] [69] [89] [73] [79] [2] [71]
JPLAG	7	[87] [57] [38] [89] [73] [79] [2]
SIM	5	[57] [38] [73] [79] [2]
Plaggie	3	[38] [73] [2]
Sherlock	3	[73] [79] [2]
NED	2	[69] [71]
SPD	2	[87] [89]

Here is a brief description of the five most used tools according to our data.

1) MOSS: It was the most used tool in the papers of this study. It supports 26 different languages and the analysis of the submitted codes is done remotely on a server at Stanford University. The codes are stored there for 14 days before being deleted. It uses a document fingerprinting algorithm called winnowing [57]. MOSS is called from the command line and the final results are displayed in HTML format [89].

2) JPLAG: Runs as a command line JAR file. The output is in HTML format, displaying the similar parts of the source codes for the user. It is multi-platform and requires only a JVM running locally. JPLAG is compatible with 5 different programming languages and can also be used with natural language text [57].

3) SIM: It was programmed in C, runs locally and without any graphical interface. It can test programs written in C, Java, Pascal, Modula-2, Lisp, Miranda, and for natural language too [57].

4) Plaggie: Compared to the other tools, there was not much information about Plaggie in our investigation. It is an open source project based on JPLAG, and compatible only with the Java language [38].

5) Sherlock: It is a stand-alone Java client, and its output is an interactive graph that displays connections between the analyzed codes and the similarity percentage. Sherlock is compatible with C++ and Java [89].

V. DISCUSSION

This section presents some comments on the results of the research questions.

We have seen an increase in the number of publications over the years. This number is expected to continue to grow in the coming years due to the increase in distance learning caused by the pandemic.

Papers are being published in a decentralized way all over the world. Researchers in several countries are addressing the plagiarism problem, but China and United States countries were prominent in our results. Also, ICSESS and ACM SIGCSE were the conferences with the most number of publications.

The publications analyzed here showed important contributions on ways to detect plagiarism. Most of the papers mapped here used solely students' source codes, however we found papers that the process of creating the source code, the user behavior in the learning management system and others.

Although Java is the most widely used language reported in this literature mapping, it is interesting to note that many articles propose generic solutions that work independently of the programming language. MOSS and JPLAG are the most used tools according to our literature mapping. These tools are versatile and suitable for assisting in plagiarism detection in general cases.

VI. CONCLUSION

In this work we presented a systematic literature mapping in the area of source code plagiarism detection in academic environments.

Our analysis has shown that the detection of plagiarism is an expanding area of study and that consistent research is being carried out worldwide. Our outcomes also pointed out the most used existing tools, as well as some new approaches to solve this problem.

The results of this investigation indicated that there is room for innovation in this field. One possibility is to analyze other aspects of the student's behavior in conjunction with the traditional comparison of source codes.

Research on new algorithms under development may be a topic for future work. Analyzing their weaknesses and strengths can support the creation of new tools for detecting plagiarism.

REFERENCES

[1] (accessed November 20, 2020). Plagiarism. https://dictionary.cambridge.org/pt/dicionario/ingles/plagiarism

[2] Matija Novak, Mike Joy, and Dragutin Kermek. 2019. Source-code Similarity Detection and Detection Tools Used in Academia. ACM Transactions on Computing Education 19, 3 (June 2019), 1–37. https://doi.org/10.1145/3313290

[3] Michal Ďuračík, Emil Kršák, and Patrik Hrkút. 2019. Searching source code fragments using incremental clustering. Concurrency and Computation: Practice and Experience 32, 13 (June 2019). https://doi.org/10.1002/cpe.5416

[4] Oscar Karnalim and Simon. 2020. Syntax Trees and Information Retrieval to Improve Code Similarity Detection. In Proceedings of the Twenty-Second Australasian Computing Education Conference. ACM. https://doi.org/10.1145/3373165.3373171

[5] Shalini Sharma, Chandra Shekhar Sharma, and Veena Tyagi. 2015. Plagiarism detection tool "parikshak". In 2015 International Conference on Communication, Information & Computing Technology (ICCICT). IEEE. https://doi.org/10.1109/iccict.2015.7045739

[6] B. Kitchenham and S. Charters. 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001. KeeleUniversity and Durham University.

[7] Nees Jan Van Eck and Ludo Waltman. 2010. Software survey: VOSviewer, a computer program for bibliometric mapping. scientometrics 84, 2 (2010), 523–538.

[8] Yuanyuan Li, Yu Sheng, Lei Xiao, and Fu Wang. 2015. A Similarity Detection Platform for Programming Learning. In Proceedings of the 7th International Conferenceon Computer Supported Education. SCITEPRESS - Science and and Technology Publications. https://doi.org/10.5220/0005490304800485

[9] Wen Wu, Xue Xiaobo, Li Ya, Gu Peng, and Xu Jianfeng. 2019. Code Similarity Detection using AST and Textual Information. International Journal of Performability Engineering 15, 10 (2019), 2683. https://doi.org/10.23940/ijpe.19.10.p14.26832691

[10] Feng Zhang, Lulu Li, Cong Liu, and Qingtian Zeng. 2020. Flow Chart Generation-Based Source Code Similarity Detection Using Process Mining. Scientific Programming 2020 (July 2020), 1–15. https://doi.org/10.1155/2020/8865413

[11] Bin Xu, Fan Gao, Kening Gao, Changkuan Zhao, and Dan Yang. 2019. Measuring code similarity using word mover's distance for programming course. In ACM International Conference Proceeding Series. ACM Press. https://doi.org/10.1145/3321408.3322862

[12] Yanyan Jiang and Chang Xu. 2018. Needle: Detecting code plagiarism on student submissions. In ACM International Conference Proceeding Series. ACM Press, 27–32. https://doi.org/10.1145/3210713.3210724

[13] Xiaohong Su, Jing Qiu, Tiantian Wang, and Lingling Zhao. 2016. Optimization and improvements of a Moodle-Based online learning system for C programming. In 2016 IEEE Frontiers in Education Conference (FIE). IEEE. https://doi.org/10.1109/fie.2016.7757699

[14] Xin Liu, Chan Xu, and Boyu Ouyang. 2015. Plagiarism Detection Algorithm for Source Code in Computer Science Education. International Journal of Distance Education Technologies 13, 4 (Oct. 2015), 29–39. https://doi.org/10.4018/ijdet.2015100102

[15] Li Qinqin and Zhang Chunhai. 2017. Research on Algorithm of Program Code Similarity Detection. In 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC). IEEE. https://doi.org/10.1109/iccsec.2017.8446728

[16] Huang Qiubo, Tang Jingdong, and Fang Guozheng. 2019. Research on Code Plagiarism Detection Model Based on Random Forest and Gradient Boosting Decision Tree. In Proceedings of the 2019 International Conference on Data Mining and Machine Learning - ICDMML 2019. ACM Press. https://doi.org/10.1145/3335656.3335692

[17] Deqiang Fu, Yanyan Xu, Haoran Yu, and Boyang Yang. 2017. WASTK: A Weighted Abstract Syntax Tree Kernel Method for Source Code Plagiarism Detection. Scientific Programming 2017 (2017), 1–8. https://doi.org/10.1155/2017/7809047

[18] Farhan Ullah, Junfeng Wang, Muhammad Farhan, Sohail Jabbar, Zhiming Wu, and Shehzad Khalid. 2018. Plagiarism detection in students' programming assignments based on semantics: multimedia e-learning based smart assessment methodology. Multimedia Tools and Applications 79, 13-14 (March 2018), 8581–8598. https://doi.org/10.1007/s11042-018-5827-6

[19] Lujiang Yu, Hui Jiang, Hongming Zhu, Qinpei Zhao, and Jessie Chen. 2020. Investigating the Understanding of Plagiarism: A Case Study of Code Plagiarismin China. In 2020 15th International Conference on Computer Science & Education(ICCSE). IEEE. https://doi.org/10.1109/iccse49874.2020.9201827

[20] Wan, H., Liu, K., and Gao, X. (2018, October). Token-based Approach for Real-time Plagiarism Detection in Digital Designs. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-5). IEEE.

[21] Huang, Q., Song, X., and Fang, G. (2020, June). Code Plagiarism Detection Method Based on Code Similarity and Student Behavior Characteristics. In 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) (pp. 167-172). IEEE.

[22] Narjes Tahaei and David C. Noelle. 2018. Automated Plagiarism Detection for Computer Programming Exercises Based on Patterns of Resubmission. In Proceedings of the 2018 ACM Conference on International Computing Education Research. ACM. https://doi.org/10.1145/3230977.3231006

[23] Tony Ohmann and Imad Rahal. 2014. Efficient clustering-based source code plagiarism detection using PIY. Knowledge and Information Systems 43, 2 (March 2014), 445–472. https://doi.org/10.1007/s10115-014-0742-2

[24] Dana Sheahen and David Joyner. 2016. TAPS: A MOSS extension for detecting software plagiarism at scale. In L@S 2016 - Proceedings of the 3rd 2016 ACMConference on Learning at Scale. ACM Press, 285–288. https://doi.org/10.1145/2876034.2893435 Cited By :5.

[25] Christian Domin, Henning Pohl, and Markus Krause. 2016. Improving Plagiarism Detection in Coding Assignments by Dynamic Removal of Common Ground. In Conference on Human Factors in Computing Systems - Proceedings. ACM Press. https://doi.org/10.1145/2851581.2892512

[26] Anish Khazane, Jia Mao, India Irish, Rocko Graziano, and Thad Starner. 2020.BELT. In Proceedings of the Seventh ACM Conference on Learning @ Scale. ACM.https://doi.org/10.1145/3386527.3406727

[27] Philip Reid Brown and Ilene J. Rosen. 2020. Misunderstandings, mistakes, and dishonesty: A post-hoc analysis of a large-scale plagiarism case in a first-year computer programming course. In ASEE Annual Conference and Exposition, Conference Proceedings, Vol. 2020-June.

[28] Breanna Devore-McDonald and Emery D. Berger. 2020. Mossad: defeating software plagiarism detection. Proceedings of the ACM on Programming Languages 4, OOPSLA (Nov. 2020), 1–28. https://doi.org/10.1145/3428206

[29] Edgcomb, A., Vahid, F., Lysecky, R., and Lysecky, S. (2017, March). Getting students to earnestly do reading, studying, and homework in an introductory programming class. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 171-176).

[30] Pierce, J., and Zilles, C. (2017, March). Investigating student plagiarism patterns and correlations to grades. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 471-476).

[31] Yan, L., McKeown, N., Sahami, M., and Piech, C. (2018, February). TMOSS: Using intermediate assignment work to understand excessive collaboration in large classes. In Proceedings of the 49th ACM technical symposium on computer science education (pp. 110-115).

[32] Mason, T., Gavrilovska, A., and Joyner, D. A. (2019, February). Collaboration versus cheating: Reducing code plagiarism in an online MS computer science program. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 1004-1010).

[33] Butler, L., Challen, G., and Xie, T. (2020, November). Data-Driven Investigation into Variants of Code Writing Questions. In 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T) (pp. 1-10). IEEE.

[34] Graziano, R., Benton, D., Wahal, S., Xue, Q., Miller, P. T., Larsen, N., ... and Starner, T. (2019, June). Jack Watson: Addressing contract cheating at scale in online computer science education. In Proceedings of the Sixth (2019) ACM Conference on Learning@ Scale (pp. 1-4).

[35] Nichols, L., Dewey, K., Emre, M., Chen, S., and Hardekopf, B. (2019, July). Syntax-based improvements to plagiarism detectors and their evaluations. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (pp. 555-561).

[36] Dimpal Shah, Heena Jethani, and Hardik Joshi. 2015. (CLSCR) cross language source code reuse detection using intermediate language. In CEUR Workshop Proceedings, Vol. 1587. 15–18. www.scopus.com Cited By :1.

[37] Mayank Agrawal and Dilip Kumar Sharma. 2016. A novel method to find out the similarity between source codes. In 2016 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON). IEEE. https://doi.org/10.1109/upcon.2016.7894676

[38] Mayank Agrawal and Dilip Kumar Sharma. 2016. A state of art on source code plagiarism detection. In 2016 2nd International Conference on Next Generation Computing Technologies (NGCT). IEEE. https://doi.org/10.1109/ngct.2016.7877421

[39] Shikha Jain, Parmeet Kaur, Mukta Goyal, and G. Dhanalekshmi. 2017. CPLAG: Efficient plagiarism detection using bitwise operations. In 2017

Tenth International Conference on Contemporary Computing (IC3). IEEE. https://doi.org/10.1109/ic3.2017.8284313

[40] Syed Mohdfazalulhaque, Vemuru Srikanth, and Edara Sreenivasa Reddy. 2017. Detection of logical clone in code using data dependency and expression list. Journal of Theoretical and Applied Information Technology 95, 12 (2017), 2666–2672. www.scopus.com

[41] Karuna Puri and Preeti Mulay. 2015. Hawk Eye: A Plagiarism Detection System. In Advances in Intelligent Systems and Computing. Springer India, 195–203. https://doi.org/10.1007/978-81-322-2517-1_20

[42] Abhishek Varat, Mayur Vetal, Pooja Bawadkar, Shubham Shinde, and Varsha Naik. 2017. Online interactive educational system for submission and evaluation of programming assignments. In 2017 International Conference on Intelligent Computing and Control (I2C2). IEEE. https://doi.org/10.1109/i2c2.2017.8321792

[43] Sharma R.Mukuntha Priya, Anukul Dixit, Krishanu Das, and Ronak Harish Patil. 2019. Plagiarism detection in source code using machine learning. International Journal of Engineering and Advanced Technology 8, 4 (2019), 897–901.

[44] Jitendra Yasaswi, Sri Kailash, Anil Chilupuri, Suresh Purini, and C. V. Jawahar. 2017. Unsupervised Learning Based Approach for Plagiarism Detection in Programming Assignments. In ACM International Conference Proceeding Series. ACMPress. https://doi.org/10.1145/3021460.3021473

[45] Mayank Agrawal, Vinod Jain, and Atul Kumar Uttam. 2020. A Novel Approach for Measurement of Source Code Similarity. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). IEEE. https://doi.org/10.1109/icrito48877.2020.9197972

[46] Sudhamani, M., and Rangarajan, L. (2019). Code similarity detection through control statement and program features. Expert Systems with Applications, 132, 63-75.

[47] Oscar Karnalim. 2017. A low-level structure-based approach for detecting source code plagiarism. IAENG International Journal of Computer Science 44, 4 (2017),501–522. www.scopus.com Cited By :9.

[48] Oscar Karnalim. 2018. An abstract method linearization for detecting source code plagiarism in object-oriented environment. In Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS. IEEE, 58–61. https://doi.org/10.1109/icsess.2017.8342863 Cited By :4.

[49] Oscar Karnalim. 2016. Detecting source code plagiarism on introductory programming course assignments using a bytecode approach. In Proceedings of 2016 International Conference on Information and Communication Technology and Systems, ICTS 2016. IEEE, 63–68. https://doi.org/10.1109/icts.2016.7910274 Cited By:16.

[50] Lisan Sulistiani and Oscar Karnalim. 2018. ES-Plag: Efficient and sensitive source code plagiarism detection tool for academic environment. Computer Applications in Engineering Education 27, 1 (Sept. 2018), 166–1822. https://doi.org/10.1002/cae.22066

[51] Oscar Karnalim and Gisela Kurniawati. 2020. Programming style on source code plagiarism and collusion detection. International Journal of Computing 19, 1(2020), 27–38.

[52] Oscar Karnalim. 2020. TF-IDF Inspired Detection for Cross-Language Source Code Plagiarism and Collusion. Computer Science 21, 1 (Jan. 2020). https://doi.org/10.7494/csci.2020.21.1.3389

[53] Oscar KARNALIM, Setia BUDI, Hapnes TOBA, and Mike JOY. 2019. SourceCode Plagiarism Detection in Academia with Information Retrieval: Dataset and the Observation. Informatics in Education 18, 2 (Oct. 2019), 321–344. https://doi.org/10.15388/infedu.2019.15

[54] Ricardo Franclinton, Oscar Karnalim, and Mewati Ayub. 2020. A Scalable Code Similarity Detection with Online Architecture and Focused Comparison for Maintaining Academic Integrity in Programming.International journal of on lineand biomedical engineering 16, 10 (2020), 40–52.

[55] Oscar Karnalim and Simon. 2020. Disguising Code to Help Students Understand Code Similarity. InKoli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research. ACM. https://doi.org/10.1145/3428029.3428064

[56] Oscar Karnalim, Simon, and William Chivers. 2020. Preprocessing for Source Code Similarity Detection in Introductory Programming. In Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research. ACM. https://doi.org/10.1145/3428029.3428065

[57] Alireza Ahadi and Luke Mathieson. 2019. A Comparison of Three Popular Source code Similarity Tools for Detecting Student Plagiarism. In ACM International Conference Proceeding Series. ACM Press, 112–117. https://doi.org/10.1145/3286960.3286974 Cited By :4.

[58] Hayden Cheers, Yuqing Lin, and Shamus P. Smith. 2019. A Novel Approach for Detecting Logic Similarity in Plagiarised Source Code. In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). IEEE. https://doi.org/10.1109/icsess47205.2019.9040752

[59] Hayden Cheers and Yuqing Lin. 2020. A Novel Graph-Based Program Representation for Java Code Plagiarism Detection. In Proceedings of the 3rd International Conference on Software Engineering and Information Management. ACM. https://doi.org/10.1145/3378936.3378960

[60] Hayden Cheers, Yuqing Lin, and Shamus P. Smith. 2020. Detecting Pervasive Source Code Plagiarism through Dynamic Program Behaviours. In Proceedings of the Twenty-Second Australasian Computing Education Conference. ACM. https://doi.org/10.1145/3373165.3373168

[61] Hayden Cheers, Yuqing Lin, and Shamus P. Smith. 2019. SPPlagiarise: A Tool for Generating Simulated Semantics-Preserving Plagiarism of Java Source Code. In 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). IEEE. https://doi.org/10.1109/icsess47205.2019.9040853

[62] Michal Duracik, Emil Krsak, and Patrik Hrkut. 2018. Issues with the Detection of Plagiarism in Programming Courses on a Larger Scale. In 2018 16th International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE. https://doi.org/10.1109/iceta.2018.8572260

[63] Michal Duracik, Emil Krsak, and Patrik Hrkut. 2018. Scalable Source Code Plagiarism Detection Using Source Code Vectors Clustering. In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). IEEE. https://doi.org/10.1109/icsess.2018.8663708

[64] Juraj Petrik, Daniela Chuda, and Branislav Steinmuller. 2017. Source code plagiarism detection: The Unix way. In 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI). IEEE. https://doi.org/10.1109/sami.2017.7880355

[65] Patrik Hrkút, Michal Ďuračík, M. Mikušová, Mauro Callejas-Cuervo, and Joanna Zukowska. 2020. Increasing K-Means Clustering Algorithm Effectivity for Using in Source Code Plagiarism Detection. Communications in Computer and Information Science, Vol. 1154 CCIS. 120–131 pages.

[66] Michal Duracik, Patrik Hrkut, Emil Krsak, and Stefan Toth. 2020. Abstract Syntax Tree Based Source Code Anti plagiarism System for Large Projects Set. IEEEAccess 8 (2020), 175347–175359. https://doi.org/10.1109/access.2020.3026422

[67] Heimo J. Jeske, Manoj Lall, and Okuthe P. Kogeda. 2018. A Real-time Plagiarism Detection Tool for Computer-based Assessments. Journal of Information Technology Education: Innovations in Practice 17 (2018), 023–035. https://doi.org/10.28945/3963

[68] Karen Bradshaw and Vongai Chindeka. 2019. Detecting Similarity in Multiprocedure Student Programs Using only Static Code Structure. In Communications in Computer and Information Science. Springer International Publishing, 211–226. https://doi.org/10.1007/978-3-030-35629-3_14

[69] Phatludi Modiba, Vreda Pieterse, and Bertram Haskins. 2016. Evaluating plagiarism detection software for introductory programming assignments. In Proceedings CSERC 2016 - Computer Science Education Research Conference. ACM Press, 37–46. https://doi.org/10.1145/2998551.2998558 Cited By :7.

[70] George Obaido, Pravesh Ranchod, and Richard Klein. 2016. Catching Plagiarists: Detectting plagiarism in student source code assignments in a virtual learning environment, Valencia Burjassot 46100 Spain IATED-INT Assoc Technology Education a Development, Lauri VolPI 6 (Ed.). INTED.

[71] Bertram Haskins and Vreda Pieterse. 2016. Lessons learnt in applying automated code plagiarism detection in an introductory programming module. INDEPENDENT JOURNAL OF TEACHING AND LEARNING 11, 1 (Jan. 2016), 69–81.

[72] Dragutin KERMEK and Matija NOVAK. 2016. Process Model Improvement for Source Code Plagiarism Detection in Student Programming Assignments. Informatics in Education 15, 1 (April 2016), 103–126. https://doi.org/10.15388/infedu.2016.06

[73] Matija Novak. 2016. Review of source-code plagiarism detection in academia. In 2016 39th International Convention on Information and Communication Technology,Electronics and Microelectronics (MIPRO). IEEE. https://doi.org/10.1109/mipro.2016.7522248

[74] Pajić, E., and Ljubović, V. (2019, May). Improving plagiarism detection using genetic algorithm. In 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 571-576). IEEE.

[75] Vítor T. Martins, Pedro Rangel Henriques, and Daniela da Cruz. 2015. An AST-based Tool, Spector, for Plagiarism Detection: The Approach, Functionality, and Implementation. In Communications in Computer and Information Science. Springer International Publishing, 153–159. https://doi.org/10.1007/978-3-319-27653-3_15

[76] Joao Paulo Barros. 2016. Assessment rules and student cheating: A set of concern sas guidelines. In 2016 2nd International Conference of the Portuguese Society for Engineering Education (CISPEE). IEEE. https://doi.org/10.1109/cispee.2016.7777723

[77] Nuno Gil Fonseca, Luis Macedo, and Antonio Jose Mendes. 2018. Using early plagiarism detection in programming classes to address the student's difficulties. In 2018 International Symposium on Computers in Education (SIIE). IEEE. https://doi.org/10.1109/siie.2018.8586700

[78] António Manso, Célio Marques, Vítor Alencar, and Paulo Santos. 2020. Plagiarism detection in algorithms - A case study using algorithmi. In CEUR Workshop Proceedings, Vol. 2733.

[79] Franca B. Allyson, Maciel L. Danilo, Soares M. Jose, and Barroso C. Giovanni. 2019. Sherlock N-overlap: Invasive Normalization and Overlap Coefficient for the Similarity Analysis Between Source Code. IEEE Trans. Comput. 68, 5 (May2019), 740–751. https://doi.org/10.1109/tc.2018.2881449

[80] Bruno Prado, Kalil Bispo, and Raul Andrade. 2018. X9: An Obfuscation Resilient Approach for Source Code Plagiarism Detection in Virtual Learning Environments. In Proceedings of the 20th International Conference on Enterprise Information Systems. SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0006668705170524

[81] de Campos, D. S., and Ferreira, D. J. (2020, October). Plagiarism detection based on blinded logical test automation results and detection of textual similarity between source codes. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-9). IEEE.

[82] Nane Kratzke. 2020. How Programming Students Trick and What JEdUnit Can Do Against It. In Communications in Computer and Information Science. Springer International Publishing, 1–25. https://doi.org/10.1007/978-3-030-58459-7_1

[83] Schneider, J., Bernstein, A., Vom Brocke, J., Damevski, K., and Shepherd, D. C. (2017). Detecting plagiarism based on the creation process. IEEE Transactions on Learning Technologies, 11(3), 348-361.

[84] Nikolai Scerbakov, Alexander Schukin, and Oleg Sabinin. 2018. Plagiarism Detection in SQL Student Assignments. In Advances in Intelligent Systems and Computing. Springer International Publishing, 110–115. https://doi.org/10.1007/978-3-319-73204-6_14

[85] Igor Andrianov, Svetlana Rzheutskaya, Alexey Sukonschikov, Dmitry Kochkin, Anatoly Shvetsov, and Arseny Sorokin. 2020. Duplicate and Plagiarism Search in Program Code Using Suffix Trees over Compiled Code. In 2020 26th Conference of Open Innovations Association (FRUCT). IEEE. https://doi.org/10.23919/fruct48808.2020.9087465

[86] Sergey Gorshkov, Maxim Nered, Eugene Ilyushin, Dmitry Namiot, and Vladimir Sukhomlin. 2020. Source Code Authorship Identification Using Tokenization and Boosting Algorithms. In Communications in Computer and Information Science. Springer International Publishing, 295–308. https://doi.org/10.1007/978-3-030-46895-8_23

[87] Marko J. Mišić, Zivojin Siustran, and Jelica Ž. Protić. 2016. A comparison of software tools for plagiarism detection in programming assignments. International Journal of Engineering Education 32, 2 (2016), 738–748. www.scopus.com Cited By :6.

[88] Ivan Pribela, Gordana Rakić, and Zoran D. Budimac. 2019. Detecting source code similarity using compression. In CEUR Workshop Proceedings, Vol. 2508. www.scopus.com

[89] Marko J. Misic, Jelica Z. Protic, and Milo V. Tomasevic. 2017. Improving source code plagiarism detection: Lessons learned. In 2017 25th Telecommunication Forum (TELFOR). IEEE. https://doi.org/10.1109/telfor.2017.8249481

[90] Olfat M. Mirza, Mike Joy, and Georgina Cosma. 2017. Suitability of BlackBox dataset for style analysis in detection of source code plagiarism. In 2017 Seventh International Conference on Innovative Computing Technology (INTECH). IEEE. https://doi.org/10.1109/intech.2017.8102424

[91] Andrés M. Bejarano, Lucy E. García, and Eduardo E. Zurek. 2013. Detection of source code similitude in academic environments. Computer Applications in Engineering Education 23, 1 (Jan. 2013), 13–22. https://doi.org/10.1002/cae.21571

[92] Krista Longi, Juho Leinonen, Henrik Nygren, Joni Salmi, Arto Klami, and Arto Vihavainen. 2015. Identification of programmers from typing patterns. In ACM International Conference Proceeding Series. ACM Press. https://doi.org/10.1145/2828959.2828960

[93] A. Omar Portillo-Dominguez, Vanessa Ayala-Rivera, Evin Murphy, and John Murphy. 2017. A unified approach to automate the usage of plagiarism detection tools in programming courses. In 2017 12th International Conference on Computer Science and Education (ICCSE). IEEE. https://doi.org/10.1109/iccse.2017.8085456

[94] Mengya Zheng, Xingyu Pan, and David Lillis. 2018. CodEX: Source code plagiarism detection based on abstract syntax trees. In CEUR Workshop Proceedings, Vol. 2259. 362–373. www.scopus.com

[95] Asako Ohno. 2015. Application of content-and-style-based source code similarity measuring methods towards programming education support system. ICICExpress Letters, Part B: Applications 6, 5 (2015), 1405–1410. www.scopus.com

[96] Hiroshi Kikuchi, Takaaki Goto, Mitsuo Wakatsuki, and Tetsuro Nishino. 2015. A Source Code Plagiarism Detecting Method Using Sequence Alignment with Abstract Syntax Tree Elements. International Journal of Software Innovation 3, 3(July 2015), 41–56. https://doi.org/10.4018/ijsi.2015070104

[97] Chiaying Lin and Chiencheng Chou. 2019. Development of an online exam platform for the programming language course: Ontology-based approach. In EG-ICE 2010 - 17th International Workshop on Intelligent Computing in Engineering. www.scopus.com

[98] Victor R. L. Shen and Farica P. Putri. 2016. A Code Plagiarism Detection System Based on Abstract Syntax Tree and a High Level Fuzzy Petri Net. In DEStech Transactions on Materials Science and Engineering. MMME.

[99] Jannatul Ferdows, Sumi Khatun, Miftahul Jannat Mokarrama, and Mohammad Shamsul Arefin. 2020. A Framework for Checking Plagiarized Contents in Programs Submitted at Online Judging Systems. In Advances in Intelligent Systems and Computing. Springer International Publishing, 546–560. https://doi.org/10.1007/978-3-030-51859-2_50

[100] Sébastien Combéfis and Arnaud Schils. 2016. Automatic programming error class identification with code plagiarism-based clustering. In Proceedings of the 2nd International Code Hunt Workshop on Educational Software Engineering - CHESE2016. ACM Press. https://doi.org/10.1145/2993270.2993271

[101] Vedran Ljubovic and Enil Pajic. 2020. Plagiarism Detection in Computer Programming Using Feature Extraction From Ultra-Fine-Grained Repositories. IEEE Access 8 (2020), 96505–96514. https://doi.org/10.1109/access.2020.2996146

[102] Lefteris Moussiades. 2016. Discovering Clusters of Plagiarism in Students' Source Codes. Journal of Engineering Science and Technology Review 9, 1 (Feb. 2016), 8–12. https://doi.org/10.25103/jestr.091.02

[103] Hamza Aldabbas. 2019. An IoT based framework for students' interaction and plagiarism detection in programming assignments. Journal of Theoretical and Applied Information Technology 97, 18 (2019), 4723–4737. www.scopus.com Cited By :1.

[104] Ahmad Shukri Mohd Noor, Farizah Yunus, Hoo Jian Liang, and Nur F. Mat Zin. 2017. Programming similarity checking system. Journal of Telecommunication, Electronic and Computer Engineering 9, 3-5 Special Issue (2017), 89–94. www.scopus.com Cited By :1.

[105] Emil Stankov, Mile Jovanov, Bojan Kostadinov, and Ana Madevska Bogdanova.2015. A new model for collaborative learning of programming using source code similarity detection. In 2015 IEEE Global Engineering Education Conference (EDUCON). IEEE. https://doi.org/10.1109/educon.2015.7096047

[106] Morten Goodwin and Tom Drange. 2016. Teaching Programming to Large Student Groups through Test Driven Development - Comparing Established Methods with Teaching based on Test Driven Development. In Proceedings of the 8th International Conference on Computer Supported Education. SCITEPRESS - Science and and Technology Publications. https://doi.org/10.5220/0005789502810288

[107] Cosmin Striletchi, Mircea Vaida, Ligia Chiorean, and Sorin Popa. 2016. A cross-platform solution for software plagiarism detection. In 2016 12th International Symposium on Electronics and Telecommunications, ISETC 2016 - Conference Proceedings. IEEE, 141–144. https://doi.org/10.1109/isetc.2016.7781077 Cited By :1.

[108] Mümine Keleş Kaya and Selma Ayşe Özel. 2014. Integrating an online compiler and a plagiarism detection tool into the Moodle distance

education system for easy assessment of programming assignments. Computer Applications in Engineering Education 23, 3 (July 2014), 363–373. https://doi.org/10.1002/cae.21606

[109] Marko Pozenel, Luka Furst, and Viljan Mahnicc. 2015. Introduction of the automated assessment of homework assignments in a university-level programming course. In 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE. https://doi.org/10.1109/mipro.2015.7160373

[110] Kihwa Lee, Yeoneo Kim, and Gyun Woo. 2019. Measuring Similarity Between Data Structures for Detecting Plagiarized Source Codes. In Proceedings of the International Conference on Data Engineering 2015 (DaEng-2015). Springer Singapore, 343–351. https://doi.org/10.1007/978-981-13-1799-6_36

[111] Enrique Flores, AAlberto Barròn-Cedeño, Lidia Moreno, and Paolo Rosso. 2014. Uncovering source code reuse in large-scale academic environments. Computer Applications in Engineering Education 23, 3 (Aug. 2014), 383–390. https://doi.org/10.1002/cae.21608

[112] Chawalit Saoban and Sunisa Rimcharoen. 2019. Identifying an original copy of the source codes in programming assignments. In 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE. https://doi.org/10.1109/jcsse.2019.8864196