

# Which Source Code Plagiarism Detection Approach is More Humane?

Oscar Karnalim

*Faculty of Information Technology  
Maranatha Christian University  
Bandung, Indonesia  
oscar.karnalim@it.maranatha.edu*

Lisan Sulistiani

*Faculty of Information Technology  
Maranatha Christian University  
Bandung, Indonesia  
lisans1601@gmail.com*

**Abstract**—This paper contributes in developing source code plagiarism detection that is more aligned with human perspective. Three evaluation mechanisms that directly relate human perspective with evaluated approaches are proposed: think-aloud, aspect-oriented, and empirical mechanism. Using those mechanisms, a comparative study toward attribute- and structure-based plagiarism detection approach (i.e., two popular approach categories in source code plagiarism detection) is conducted. According to that study, structure-based approach is more effective than the attribute-based one; its signature aspect and resulted similarity degrees are more related to human preferences. In addition, such approach is related to most human-oriented aspects for suspecting source code plagiarism.

**Keywords**—source code plagiarism detection; human-oriented evaluation; programming; human-computer cooperation; computer science education

## I. INTRODUCTION

Source code plagiarism occurs when a student claims another student's source code as their original work [1]. In programming courses, it is considered as an illegal behavior; the correlation between student's knowledge and grade will be weakened, resulting more complicated recruitment process for industry. Mitigating such illegal behavior is a crucial task for programming examiners (e.g., lecturers or assistant lecturers).

To mitigate human effort, automated plagiarism detection approaches have been developed [2]. Using such approaches, plagiarism-suspected pairs could be detected in a no time. However, most approaches (where some of them are implemented on publicly available tools such as JPlag [3]) could be either too sensitive or insensitive when compared to human. Some of them could generate numerous false positives (i.e., non-plagiarized pairs which are considered as plagiarism-suspected pairs) while the others could generate numerous false negatives (i.e., plagiarized pairs which are not considered as plagiarism-suspected pairs). The existence of false results would complicate accusation process: examiner needs to double-check each result comprehensively to avoid mis-detection.

One of the solutions for reducing false results is to utilize a plagiarism detection approach which behaves like humans (i.e., examiners), considering correct detection results are originally defined by them. Therefore, it is natural to evaluate plagiarism

detection approaches from human perspective. Nevertheless, to the best of our knowledge, existing evaluation mechanisms only indirectly consider human perspective. We would argue that direct relation between human perspective and evaluated plagiarism detection approaches are needed to favor human-like approach in evaluation.

This paper proposes two contributions. First, three evaluation mechanisms that directly relate human perspective to evaluated plagiarism detection approaches are proposed. Such mechanisms could become evaluation alternatives in addition to conventional mechanism (which relies on human-annotated dataset). Second, using proposed evaluation mechanisms, a comparative study between attribute- and structure-based approach (i.e., two well-known approach categories in plagiarism detection [4], [5]) is presented. This study draws the applicability of proposed evaluation mechanisms.

## II. RELATED WORKS

In general, source code plagiarism detection approaches work in two phases: conversion and comparison phase. Conversion phase translates given source codes and/or their features to intermediate representations for more effective and efficient comparison phase. Some of frequently-used intermediate representations are: source code token sequence [3], [4], [6]–[19], source code word sequence (i.e., a sequence generated from natural language tokenizer) [20], [21], low-level token sequence (i.e., a sequence generated from executable file of given source code) [5], [22]–[25], program dependency graph [26], and abstract syntax tree [27], [28]. After translated, resulted intermediate representations are then passed to comparison phase—that measures similarity degree and defines which pairs are suspected as plagiarism cases. With regard to this phase, plagiarism detection approaches can be further classified into three categories: attribute-based, structure-based, and hybrid approach [4], [5].

Attribute-based approach determines plagiarism-suspected pairs based on shared source code characteristics (e.g., the number of unique operators [8] and token occurrences [10]). Initially, similarity measurement on such approach relies on attribute-counting mechanism; two codes are considered similar if both of them share the same characteristic occurrence

frequencies [8], [9], [29]. Later, advanced similarity measurements are incorporated. These measurements are adapted from Fuzzy Logic (e.g., Fuzzy C-Means [10]), Machine Learning (e.g., Random Forest [28]), and Information Retrieval (e.g., Cosine Similarity [11], [12], Jaccard Coefficient [14], Language Model [28], and Latent Semantic Analysis [20], [21]).

Structure-based approach determines plagiarism-suspected pairs based on shared source code structure. This approach typically relies on string-matching algorithms—such as Running-Karp-Rabin Greedy-String-Tiling [3], [5], [17]–[19], [22], [24], [25], local alignment [23], and Winowing algorithm [17]—that have been modified to handle token streams. However, considering the limitation of those algorithms, some graph-matching algorithms (e.g., tree kernel algorithm [27] and graph isomorphism algorithm [26]) are proposed as alternative comparison algorithms.

Hybrid approach combines both attribute- and structure-based approach for determining plagiarism-suspected pairs. Such combination aims to enhance either effectiveness [6], [7], [16] or efficiency [15]. For enhancing effectiveness, these combined approaches either show the results of both conventional approaches at once [7] or treat the result of one approach as an input of another (e.g., using the result of structure-based approach as an attribute for learning algorithm [6] or clustering algorithm [16]). On the other, for enhancing efficiency, plagiarism-suspected pairs from attribute-based approach are used as the inputs of structure-based approach [15]. In such manner, not all source pairs are compared using structure-based approach (which is typically costly in terms of processing time).

In the context of effectiveness evaluation, two mechanisms are frequently used in source code plagiarism detection: statistical and domain-specific mechanism. Statistical mechanism utilizes metrics commonly used in statistic (such as precision) to determine the performance of given plagiarism detection approaches [3], [4], [6], [10]–[15], [17], [19]–[21], [27], [28]. This mechanism requires an evaluation dataset covering all statistical result categories (i.e., true positive, true negative, false positive, and false negative). In contrast, domain-specific mechanism utilizes metrics specifically designed for plagiarism detection (e.g., similarity degree) to determine plagiarism detection approaches' performance [5], [7], [16], [18], [22]–[26], [30]. Unique to this mechanism, it only relies on true positives (that are frequently created in controlled environment). In other words, it is more practical to be used (since only true positives are required) yet it is less related to real environment (since other result categories are not considered).

One of the objectives in developing automated source code plagiarism detection is to let computer imitates manual plagiarism detection done by human. Hence, it is natural to evaluate the effectiveness of plagiarism detection approaches from human perspective. However, existing evaluation mechanisms (i.e., statistical and domain-specific mechanism) may not sufficiently reflect human perspective; considering such perspective is not directly related to evaluated approaches; they are only related in indirect manner through human-annotated

dataset. Such dissonance could complicate accusation process due to the existence of false results.

### III. HUMAN-ORIENTED EVALUATION MECHANISMS

Three human-oriented evaluation mechanisms (i.e., evaluation mechanisms that directly relate human perspective with evaluated plagiarism detection approaches) are proposed: think-aloud, aspect-oriented, and empirical evaluation mechanism. To mitigate qualitative bias, human-oriented effectiveness is determined by comparing plagiarism detection approaches to each other through questionnaire survey. Considering such comparison is best performed with only two instances, proposed evaluation mechanisms are designed to accept two approaches per execution. However, if necessary, they could compare numerous approaches by adapting tournament selection from Genetic Algorithm [31] (with an assumption that each approach refers to one instance and fitness function is the human-oriented effectiveness degree).

#### A. Think-Aloud Evaluation Mechanism

Think-aloud evaluation mechanism compares the effectiveness of two source code plagiarism detection approaches based on respondents' descriptions about how manual detection works. It is suitable to be used when only a limited number of plagiarism cases exists for evaluation and the main goal is to observe from process perspective (i.e., factors considered while detecting plagiarism). This mechanism is inspired from think-aloud protocols used in Empirical Software Engineering [32].

Think-aloud evaluation mechanism consists of three phases. First of all, respondents are asked to manually suspect plagiarized source codes for reminding them how manual detection works. Second, they should describe how they suspected plagiarized codes as detail as possible in natural language sentences. Third, each aspect from resulted descriptions will be qualitatively linked to evaluated plagiarism detection approaches; an approach is more effective than another if its characteristics are frequently mentioned on respondents' descriptions. To mitigate bias, no clue should be given to respondents in regard to evaluated plagiarism detection approaches.

It is important to note that this mechanism might not capture some human-oriented aspects due to human limitations namely low interpretation skill and awareness. Low interpretation skill occurs when the respondent cannot formalize what is in their mind while suspecting plagiarized source codes; whereas, low awareness occurs when the respondent does not aware about some parts of detection process (which have been unconsciously conducted).

#### B. Aspect-Oriented Evaluation Mechanism

Aspect-oriented evaluation mechanism relies on one differentiating aspect between two source code plagiarism detection approaches to measure human-oriented effectiveness. If that aspect affects (or positively correlates more with) humans' preferences, it can be stated that an approach with such aspect

is more effective. Otherwise, the approach without such aspect is preferred. This mechanism is suitable to be used when there is a salient differentiating aspect between both plagiarism detection approaches and the main goal is to observe from result perspective (i.e., resulted similarity degree). In general, this mechanism works by asking respondents to rank plagiarism cases from human perspective wherein the results will be further analyzed in qualitative and quantitative perspective. It is inspired from ranking concept in Information Retrieval [33] where relevancy (i.e., similarity in our case) is defined based on ranks.

Prior to asking respondents, survey cases (that accentuate the impact of differentiating aspect) should be artificially developed. For each case, an original source code is plagiarized to several codes where the only modification is the differentiating aspect with various degree. To guarantee that the impact of differentiating aspect is shown, on such cases, the difference between both evaluated approaches in terms of similarity degree should be statistically significant. The significance itself can be measured using t-test.

For each survey case, respondents are asked to rank plagiarized codes in descending order based on their similarity degree toward original code from human perspective. Each plagiarized code will be assigned with a rank where the first rank refers to the highest similarity; if two or more codes are equally similar, they could be assigned with similar rank. To provide more objective results, each survey case should be rated by more than one respondent.

After all survey cases are ranked by the respondents, human-oriented effectiveness of evaluated approaches can be analyzed from both qualitative and quantitative perspective. From qualitative perspective, an approach with aforementioned differentiating aspect is more effective if human-assigned ranks are changed as the degree of involved differentiating aspect is modified. In contrast, from quantitative perspective, the effectiveness of evaluated plagiarism detection approaches is defined through Pearson correlation between similarity degrees and averaged human-assigned ranks; a plagiarism detection approach is more effective than another if its resulted similarity degrees (measured on artificially-plagiarized codes toward their originals) generate higher correlation toward averaged human-assigned ranks. Considering similarity degree's result interpretation (where higher value is preferred) contradicts human-assigned rank's (where lower value is preferred), averaged human-assigned ranks will be negated before correlating to make its result interpretation proportional to similarity degree's.

### C. Empirical Evaluation Mechanism

Empirical evaluation mechanism relies on humans' preferences toward contradicting plagiarism pairs (i.e., pairs where each of their members exclusively favors a plagiarism detection approach), selected from existing plagiarism dataset, to evaluate human-oriented effectiveness. This mechanism is suitable to be used when the main goal is to observe from result perspective (i.e., resulted similarity degree) and a

publicly-available plagiarism dataset is involved. It is inspired from ranking concept in Information Retrieval [33] (where relevancy is defined based on ranks) and A/B testing in Web Analytics [34] (where human should select one between two instances based on their qualitative perspective).

Contradicting plagiarism pairs are selected from a plagiarism dataset in fourfold. First, plagiarized codes are clustered based on their original code (one cluster for each original code). Second, per plagiarism detection approach, plagiarized codes will be compared with their originals, resulting a set of similarity degrees. Third, for each cluster and a plagiarism detection approach, plagiarized codes are ranked in descending similarity order. Fourth, contradicting plagiarism pairs are selected for each cluster. Two plagiarized source codes (e.g., *A* and *B*) are considered as a contradicting plagiarism pair *iff* returned value from either (1) or (2) is true. *P1* returns a rank resulted from the first plagiarism detection approach while *P2* returns a rank resulted from another.

$$Con1(A, B) = (P1(A) > P1(B)) \wedge (P2(A) < P2(B)) \quad (1)$$

$$Con2(A, B) = (P1(A) < P1(B)) \wedge (P2(A) > P2(B)) \quad (2)$$

To provide more insight, suppose we have three plagiarized codes (*A*, *B*, and *C*); their similarity degree toward original code—measured using two detection approaches: *P1* and *P2*—can be seen on Table I. According to given example, *P1*'s and *P2*'s ranking order are {*A*, *C*, *B*} and {*C*, *A*, *B*} respectively. As a result, *A* and *C* are considered as a contradicting pair; they satisfy (1) since  $P1(A) > P1(C)$  and  $P2(A) < P2(C)$ .

TABLE I  
SIMILARITY DEGREES FOR ILLUSTRATING HOW CONTRADICTING  
PLAGIARISM PAIRS ARE SELECTED

Plagiarized Code	P1's Similarity Degree	P2's Similarity Degree
A	70%	50%
B	50%	40%
C	60%	95%

Two things should be considered while selecting contradicting pairs. First, plagiarized codes in each contradicting pair should be explicitly different to each other when perceived by human. Hence, it is preferable to select pairs that generate high delta similarity degree between evaluated approaches. Second, the difference between evaluated approaches should not be coincidental on selected contradicting pairs. Hence, it is important to assure that similarity degrees resulted from both approaches are significantly different to each other; wherein the significance can be measured using t-test.

For each contradicting pair, each respondent should select one plagiarized code that seems more similar to original code; preferred code will be assigned with the 1<sup>st</sup> rank while another will be assigned with the 2<sup>nd</sup> rank.

The result of this evaluation can be quantitatively analyzed by correlating averaged human-assigned ranks with resulted similarity degrees (measured on plagiarized codes with their originals per evaluated plagiarism detection approach) using

Pearson correlation; higher correlation refers to higher effectiveness degree. Similar to aspect-oriented mechanism, averaged human-assigned ranks should be negated beforehand to make its result interpretation proportional to similarity degree's (where higher value is preferred). Further, the effectiveness of evaluated plagiarism detection approaches can also be naively measured by counting how many favoring codes are assigned with the 1<sup>st</sup> rank for each approach.

Qualitative analysis is discouraged to be conducted. Plagiarized codes are not specifically designed to accentuate the difference between evaluated plagiarism detection approaches. It will therefore be difficult to qualitatively collect findings regarding why a favoring code is preferred than another.

#### IV. CASE STUDY: COMPARING ATTRIBUTE- WITH STRUCTURE-BASED APPROACH

As our case study, proposed evaluation mechanisms will be applied to compare human-oriented effectiveness of Attribute-Based Approach (ABA) and Structure-Based Approach (SBA). These approaches are the baseline of most plagiarism detection approaches [4], [5]. We are aware that a work proposed in [35] also compares those categories. Yet, human perspective is not directly linked to plagiarism detection approaches in their evaluation.

Both ABA and SBA work by accepting two source codes to return a similarity degree. At first, inputted source codes are tokenized using ANTLR [36] (with an exclusion of comment tokens). Later, resulted token sequences will be compared to each other using specific similarity algorithm. ABA will utilize Vector Space Model and Cosine Similarity [33] (where each vector refers to token occurrence frequency). Whereas, SBA will utilize Running-Karp-Rabin Greedy-String-Tiling algorithm [37], with two as its minimum matching threshold and average similarity equation [38] as its normalization technique.

Fifteen lecturer assistants are involved in this evaluation. They are experienced in terms of checking students' programming assignments. Hence, it is expected that resulted evaluation findings will be considerably objective. It is true that the number of respondents are somewhat limited. However, since the main goal of this case study is to prove the applicability of proposed evaluation mechanism, we would argue such limited number of respondents are enough.

##### A. Methodology

Proposed evaluation mechanisms will be merged as one in this case study. Fig. 1 shows that such combination generates five phases: artificial cases generation, contradicting plagiarism pairs selection, respondent-answering, describing, and analysis phase.

Artificial cases (for aspect-oriented evaluation mechanism) are generated by considering token order as a differentiating aspect; such aspect is only considered by SBA while determining similarity. Four artificial cases are considered in this study, covering various scopes of order-changed token sequences: single-instruction, multiple-instructions, method, and class scope. All of them are written in Java programming

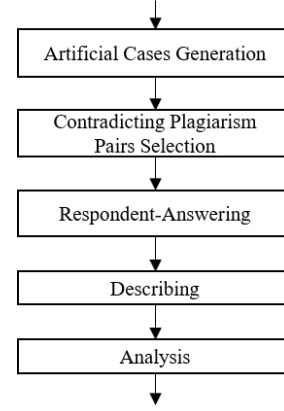


Fig. 1. Combined phases from proposed evaluation mechanisms

language. A case with single-instruction scope has 4 plagiarized codes that are formed by swapping  $N$  instructions (where  $N$  in these codes are 0, 1, 3, and 5 respectively). Remaining cases contain 6 plagiarized codes each, that are formed by changing the order of three subsequences in combinatoric manner. Generated artificial cases are valid to be used in our evaluation; their SBA's and ABA's result are statistically different (in terms of similarity degree) when measured using two-tailed paired t-test. It generates  $p=0.02$ , which is lower than 0.05.

Contradicting plagiarism pairs (for empirical evaluation mechanism) are selected from a Java-based plagiarism dataset [30] by clustering plagiarized codes per original code and plagiarism level [9], resulting 42 clusters (plagiarism level is added to our consideration for providing more comprehensive findings). Forty-five contradicting pairs are selected for our survey; they are formed from level-2 to level-6 plagiarism categories where each category contributes 5, 9, 11, 10, 10 contradicting pairs respectively. Those pairs lead to a statistically significant difference between ABA and SBA when measured using two-tailed paired t-test. It generates  $p\text{-value}=1.59\text{E-}78$ , which is lower than 0.05. Hence, they are valid to be used in this evaluation.

Regarding artificial cases and contradicting plagiarism pairs, respondents will be asked to rank them in similar manner as defined in aspect-oriented and empirical evaluation mechanism. However, considering ranking 45 contradicting pairs (for empirical evaluation mechanism) at once may generate biased result due to human fatigue, our respondents are divided to three groups of five where the member of each group is only responsible to rank one-third of total pairs.

After ranking artificial cases and contradicting plagiarism pairs, describing phase will be conducted. At this phase, respondents will be asked to write down their manual detection technique as detail as possible.

Survey results collected from respondent-answering and describing phase will be further analyzed in qualitative and quantitative manner. These analysis works exactly similar as in our proposed evaluation mechanisms.

## B. Results

According to our evaluation, token order (which is SBA's signature aspect) is considered by respondents when detecting plagiarism cases. Plagiarized code without such change (i.e., a verbatim copy of original code) is always assigned with the highest rank when compared to its counterpart with such change. Further, resulted negated ranks are strongly correlated with SBA's similarity degrees in positive manner; their correlation degree is 0.833 when measured using Pearson correlation (where 1 represents the strongest positive correlation). Consequently, it can be stated that SBA is preferred than ABA from aspect-oriented evaluation perspective.

It is important to note that the correlation between negated ranks and ABA's similarity degrees is immeasurable due to Pearson correlation's nature. It cannot correlate two sequences when one of them has no variability (which is ABA in our case study, it generates resemblant similarity degrees for all artificial cases).

When perceived from empirical evaluation, SBA is preferred than ABA in both general and plagiarism level perspective. Fig. 2 shows that SBA's percentage of respondent-preferred codes is far higher than ABA's. Further, SBA's similarity degrees are more positively correlated with negated respondent-assigned ranks despite their low correlation. As seen in Fig. 3, on all categories, SBA generates positive correlation while ABA generates the negative one. Low correlation between similarity degrees and respondent-assigned ranks is natural considering plagiarized codes are not specifically designed to accentuate the difference between evaluated plagiarism detection approaches.

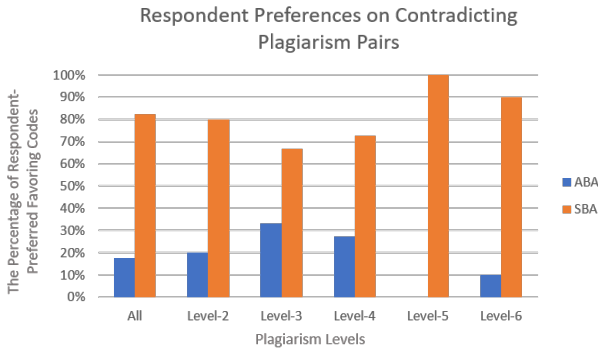


Fig. 2. The percentage of respondent-preferred favoring codes on contradicting plagiarism pairs. Each number is generated by counting how many favoring codes are assigned with the 1<sup>st</sup> rank and then normalized based on the number of involved plagiarism pairs.

SBA generates the strongest correlation on level-2 plagiarism level category (which is about identifier renaming). Hence, it can be stated that respondents tend to detect plagiarism as SBA when source code structure is unchanged. In contrast, it generates the weakest correlation on level-4 plagiarism level category. In other words, method structure change (i.e., level-4 signature attack) makes respondents focus less on source code structure.

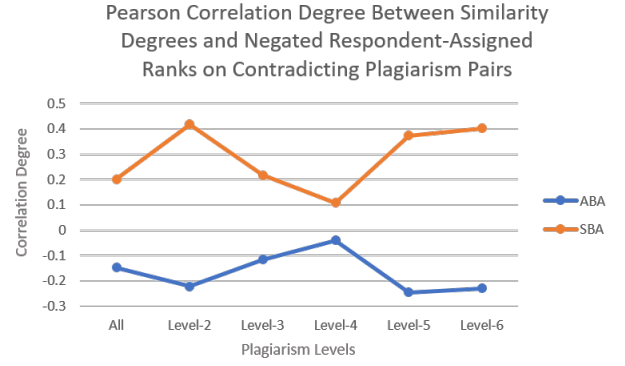


Fig. 3. Pearson correlation degree between similarity degrees and negated respondent-assigned ranks on contradicting plagiarism pairs.

Respondents' descriptions about their manual detection technique show that six aspects are considered while suspecting plagiarism cases. Table II displays those aspects, including their occurrences (calculated based on the number of respondents mentioning it) and their relationship to evaluated plagiarism detection approaches. Most of mentioned aspects are related to SBA. Hence, it can be stated that SBA is more effective than ABA from think-aloud evaluation perspective.

TABLE II  
CONSIDERED ASPECTS WHILE SUSPECTING PLAGIARISM CASES

Aspect	Occurrences	Relationship to Evaluated Approaches
Statement order	11	Related to SBA since order is a part of source code structure.
Semantic	5	Related to SBA since semantic is preserved based on source code structure.
Identifier name	3	Related to ABA since identifier name is a source code characteristic.
Structure	2	Obviously related to SBA.
Output	1	Related to SBA since output is defined based on source code structure.
Line of code	1	Related to ABA since line of code is a source code characteristic.

To sum up, from human perspective, SBA is more effective than ABA according to three rationales. First, SBA's signature aspect (i.e., token order) is considered by respondents when detecting plagiarism cases. Second, SBA's favoring cases is preferred than ABA's. Third, SBA is related to more human-oriented aspects for suspecting source code plagiarism.

## V. CONCLUSION AND FUTURE WORK

In this paper, three human-oriented evaluation mechanisms in source code plagiarism detection have been proposed. Unique to these mechanisms, they directly relate human perspective with evaluated plagiarism detection approaches. These mechanisms have been tested on a case study toward attribute- and structure-based plagiarism detection approach. According to such test, structure-based approach is more effective; its signature aspect and similarity degree is more related to respondents' preferences. Further, it is related to

most aspects that are mentioned by respondents for suspecting plagiarism cases.

For future works, we plan to evaluate human-oriented effectiveness of frequently-used features on source code plagiarism detection (such as method linearization [24], [25]) using aspect-oriented evaluation mechanism. Further, we also plan to systematize how human suspects source code plagiarism (with and without machine learning algorithm). The results of such work (e.g., features with high human-oriented effectiveness degree and systematic steps to suspect plagiarism) will be used to develop source code plagiarism detection approach that is more aligned to human perspective.

## REFERENCES

- [1] G. Cosma and M. Joy, "Towards a Definition of Source-Code Plagiarism," *IEEE Transactions on Education*, vol. 51, no. 2, pp. 195–200, may 2008.
- [2] T. Lancaster and F. Culwin, "A Comparison of Source Code Plagiarism Detection Engines," *Computer Science Education*, vol. 14, no. 2, pp. 101–112, jun 2004.
- [3] L. Prechelt, G. Malpohl, and M. Philippsen, "Finding Plagiarisms among a Set of Programs with JPlag," *Journal of Universal Computer Science*, vol. 8, no. 11, pp. 1016–1038, 2002.
- [4] Z. A. Al-Khanjari, J. A. Fiaidhi, R. A. Al-Hinai, and N. S. Kutti, "PlagDetect: a Java programming plagiarism detection tool," *ACM Inroads*, vol. 1, no. 4, p. 66, dec 2010.
- [5] O. Karnalim, "A Low-Level Structure-based Approach for Detecting Source Code Plagiarism," *IAENG International Journal of Computer Science*, vol. 44, no. 4, 2017.
- [6] S. Engels, V. Lakshmanan, M. Craig, S. Engels, V. Lakshmanan, and M. Craig, "Plagiarism detection using feature-based neural networks," *ACM SIGCSE Bulletin*, vol. 39, no. 1, p. 34, mar 2007.
- [7] M. El Bachir Menai and N. S. Al-Hassoun, "Similarity detection in Java programming assignments," in *2010 5th International Conference on Computer Science & Education*. IEEE, aug 2010, pp. 356–361.
- [8] K. J. Ottenstein and K. J., "An algorithmic approach to the detection and prevention of plagiarism," *ACM SIGCSE Bulletin*, vol. 8, no. 4, pp. 30–41, dec 1976.
- [9] S. K. Faidhi, J. A. W. Robinson, "An empirical approach for detecting program similarity and plagiarism within a university programming environment," *Computers & Education*, vol. 11, no. 1, pp. 11–19, jan 1987.
- [10] G. Acampora and G. Cosma, "A Fuzzy-based approach to programming language independent source-code plagiarism detection," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, aug 2015, pp. 1–8.
- [11] U. Inoue and S. Wada, "Detecting plagiarisms in elementary programming courses," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. IEEE, may 2012, pp. 2308–2312.
- [12] T. Ohmann and I. Rahal, "Efficient clustering-based source code plagiarism detection using PIY," *Knowledge and Information Systems*, vol. 43, no. 2, pp. 445–472, may 2015.
- [13] M. Mozgovoy, K. Fredriksson, D. White, M. Joy, and E. Sutinen, "Fast Plagiarism Detection System," in *International Symposium on String Processing and Information Retrieval*. Springer, Berlin, Heidelberg, 2005, pp. 267–270.
- [14] L. Moussiades and A. Vakali, "PDetect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets," *The Computer Journal*, vol. 48, no. 6, pp. 651–661, nov 2005.
- [15] S. Burrows, S. M. M. Tahaghoghi, and J. Zobel, "Efficient plagiarism detection for large code repositories," *Software: Practice and Experience*, vol. 37, no. 2, pp. 151–175, feb 2007.
- [16] J. Y. Poon, K. Sugiyama, Y. F. Tan, and M.-Y. Kan, "Instructor-centric source code plagiarism detection and plagiarism corpus," in *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE '12*. New York, New York, USA: ACM Press, 2012, p. 122.
- [17] Z. Duric and D. Gasevic, "A Source Code Similarity System for Plagiarism Detection," *The Computer Journal*, vol. 56, no. 1, pp. 70–86, jan 2013.
- [18] A. M. Bejarano, L. E. García, and E. E. Zurek, "Detection of source code similitude in academic environments," *Computer Applications in Engineering Education*, vol. 23, no. 1, pp. 13–22, jan 2015.
- [19] C. Kustanto and I. Liem, "Automatic Source Code Plagiarism Detection," in *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*. IEEE, 2009, pp. 481–486.
- [20] E. Flores, A. Barrón-Cedeño, L. Moreno, and P. Rosso, "Cross-Language Source Code Re-Use Detection Using Latent Semantic Analysis," *Journal of Universal Computer Science*, vol. 21, no. 13, pp. 1708–1725, 2015.
- [21] G. Cosma and M. Joy, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 379–394, mar 2012.
- [22] O. Karnalim, "Detecting source code plagiarism on introductory programming course assignments using a bytecode approach," in *The 10th International Conference on Information & Communication Technology and Systems (ICTS)*. Surabaya: IEEE, 2016, pp. 63–68.
- [23] F. S. Rabbani and O. Karnalim, "Detecting Source Code Plagiarism on .NET Programming Languages using Low-level Representation and Adaptive Local Alignment," *Journal of Information and Organizational Sciences*, vol. 41, no. 1, pp. 105–123, jun 2017.
- [24] O. Karnalim, "An Abstract Method Linearization for Detecting Source Code Plagiarism in Object-Oriented Environment," in *The 8th International Conference on Software Engineering and Service Science (ICSESS)*. Beijing: IEEE, 2017.
- [25] O. Karnalim, "IR-based technique for linearizing abstract method invocation in plagiarism-suspected source code pair," *Journal of King Saud University - Computer and Information Sciences*, feb 2018.
- [26] C. Liu, C. Chen, J. Han, and P. S. Yu, "GPLAG: detection of software plagiarism by program dependence graph analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*. New York, New York, USA: ACM Press, 2006, p. 872.
- [27] D. Fu, Y. Xu, H. Yu, and B. Yang, "WASTK: A Weighted Abstract Syntax Tree Kernel Method for Source Code Plagiarism Detection," *Scientific Programming*, vol. 2017, pp. 1–8, feb 2017.
- [28] D. Ganguly, G. J. F. Jones, A. Ramírez-de-la Cruz, G. Ramírez-de-la Rosa, and E. Villatoro-Tello, "Retrieving and classifying instances of source code plagiarism," *Information Retrieval Journal*, pp. 1–23, sep 2017.
- [29] P. Vamplew and J. Dermoudy, "An anti-plagiarism editor for software development courses," in *Proceedings of the 7th Australasian conference on Computing education*. ACM, 2010, pp. 83–90.
- [30] O. Karnalim and S. Budi, "The Effectiveness of Low-Level Structure-based Approach Toward Source Code Plagiarism Level Taxonomy," in *The 6th International Conference on Information and Communication Technology (ICoICT)*. Bandung: IEEE, 2018.
- [31] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall, 2003, vol. 2, no. 9.
- [32] J. Singer, S. E. Sim, and T. C. Lethbridge, "Software Engineering Data Collection for Field Studies," in *Guide to Advanced Empirical Software Engineering*. London: Springer London, 2008, pp. 9–34.
- [33] W. B. Croft, D. Metzler, and T. Strohman, *Search engines : information retrieval in practice*. Addison-Wesley, 2010.
- [34] R. Kohavi and R. Longbotham, "Online Controlled Experiments and A/B Testing," in *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, 2017, pp. 922–929.
- [35] K. L. Verco and M. J. Wise, "Software for detecting suspected plagiarism," in *Proceedings of the first Australasian conference on Computer science education - ACSE '96*. New York, New York, USA: ACM Press, 1996, pp. 81–88.
- [36] T. Parr, *The definitive ANTLR 4 reference*. Pragmatic Bookshelf, 2013.
- [37] M. J. Wise, "Neweyes: A System for Comparing Biological Sequences Using the Running Karp-Rabin Greedy String-Tiling Algorithm," in *International Conference on Intelligent Systems for Molecular Biology*. AAAI, 1995.
- [38] M. J. Wise, "Detection of Similarities in Student Programs: YAP'ing May be Preferable to Plague'ing," in *Proceedings of the twenty-third SIGCSE technical symposium on Computer science education - SIGCSE '92*, vol. 24, no. 1. New York, New York, USA: ACM Press, 1992, pp. 268–271.