INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Analysis of Research Trends Towards Types of Code Clone Detection Techniques

Kamna Solanki[1]*, **Sandeep Dalal**[1]

1 Associate Professor, Maharshi Dayanand University, Rohtak, India

OPEN ACCESS

Received: 18-11-2022

Accepted: 28-12-2022

Published: 20-02-2023

Citation: Solanki K, Dalal S (2023) Analysis of Research Trends Towards Types of Code Clone Detection Techniques. Indian Journal of Science and Technology 16(7): 468-475. https://doi.org/10.17485/IJST/v16i7.2219

* **Corresponding author**.

kamna.mdurohtak@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: The key research objectives of this study are: (1.) To compare and contrast the research trend towards the tree, token, text, metric, and graph-based code clone detection techniques; (2.) To study the distribution of metric-based code clone detection techniques on various online repositories; (3.) To make a statistical analysis of the hybrid techniques available for clone detection. The overall objective is to investigate the research trends of code clone detection approaches. **Methods**: Various repositories like google scholar, IEEE, and ELSEVIER Digital Libraries were systematically examined to attain the results in terms of research articles published in various places like conferences, journals, etc. followed by the inclusion and exclusion criteria. **Findings:** (1.) The findings related to objective 1 depicted that 50% of total clone detection techniques are tree and graph-based Code Clone Detection techniques followed by 20% of text-based and 30% of token-based code-clone detection techniques (2.) The findings related to the second objective depicted that an equal percentage of 46% of research work related to metric-based code clone detection techniques has been published in journals and conferences. (3.) The findings related to the third objective showed that 43% of hybrid code clone detection techniques are based on machine learning techniques, 24% are based on neural networks, and 18% of techniques are data mining based followed by 15% nature inspired based algorithms. **Novelty:** The study conducted is novel in identifying and exploring those potential code clone detection techniques that are underutilized and least explored. The result of research questions will assist researchers to draw inferences regarding usage, application, research trends, future needs, and research directions.

Keywords: Code Clones; Clone Detection Techniques; Metric Based Clone Detection; Types of Clone Detection Techniques; Software Clones

1 Introduction

Software engineers directly copy and paste a piece of source code from another piece of source code, even with slight changes to make them identical or indistinguishable. This process is known as software/code cloning or sometimes code replication. Software

engineers usually do code cloning to accomplish their responsibilities faster. This behavior of this kind poses programming and maintenance concerns. For example, a bug detected in a cloned code segment of a software system requires developers to locate and fix the bug everywhere, increasing the complexities of maintaining software architecture [1,2]. In addition, cloning a vulnerable code can spread the vulnerability in the system as far as the security of the software framework is concerned.

1.1 Code Clone Detection

Code-clone detection refers to the operation of discovering identical or parallel segments of code in an application. Such activity greatly enables software development but also entails bug duplication. There are four main categories of code clones, type 1 to type 4, based on their resemblance. When there is a bug in the original code, its duplicate code is expected to contain a similar bug, causing the bugs to spread all over the software system, the same as an infection [3]. Even though fixing the original bug is feasible, the fixing process generally finds it difficult to treat all the code fragments cloned from the original code. To deal with issues, code clone detection-based bug detection techniques have been broadly considered and have yielded high-quality results. Therefore, code clone detection is very important as an underlying analysis technique to maintain the quality of software [1–3].

1.2 Code Clone Detection Process

The main element of code clone detection frameworks is the code clone detector. Its main objective is to obtain copy-paste or duplicate source code and process the crucial steps of clone detection. Pre-processing code is the first step in clone detection removing all redundant or inappropriate fragments of the source code, including whitespace and comments [1,2]. The next process of conversion involves converting the source code obtained from the pre-processing stage into the respective intermediary depiction for additional comparison. The detection matching step identifies similar source code fragments by comparing the source code units with the target files through a special comparison algorithm. This step generates output in the form of a list of clone pairs or cloned classes. Formatting aims to format the list of clone pairs achieved from the earlier step depending on the comparison algorithm into a fresh clone pair list corresponding to the real source code. Post-processing, also known as filtering/manual analysis, is an optional step in most code clone detection frameworks. This step filters out false positives or missed clones based on reanalysis by human experts or automatic heuristics. Clone results examined and established by earlier detection steps can be reported to the framework for more actions, for example correcting or deleting the source code [2].

1.3 Code Clone Detection Techniques

The pipeline of code clone detection techniques includes five methods: textual methods, token-based methods, syntactical methods, semantic methods, and learning methods. The first method compares two code segments utilizing text/strings/lexemes and finds clones only when the two code fragments are almost similar in the context of text content. The token-based methods bifurcate all source code lines into a series of tokens throughout the lexical analytic step of the compiler. Next, all tokens are reconverted into token series lines [3]. The token series rows are matched to locate and inform duplicate codes. There are two types of syntactical methods: tree-supported methods and metric-supported methods. Tree-based methods explore similar regions to compare sub-trees through the extraction of the AST. These identical regions form a code clone. The metric-supported clone detection methods generate separate vectors for every code piece, using metrics collected from the source code [3,4]. The areas with identical codes are discovered by comparing such vectors. A semantic method traces two pieces of code performing a similar computation but with differently designed code. Semantic code-clone detection includes a variety of approaches; However, one of the major ways is using graph-supported methods. Learning methods in the code-clone detection domain often differ greatly. The learning methods depend on machine learning or other learning schemes to detect code clones.

1.4 Related Work

Researchers have conducted diverse studies and presented several code clone detection techniques for improving the process of detecting code cloning [4–20]. There are distinct domains for these techniques. Diverse techniques for clone detection are reviewed in this section [4–12].

Yuan, et al. reviewed a new graph representation technique based on intermediate code for detecting functional code clones. Subsequently, the Softmax classification algorithm for detecting the functional code clone pairs. The bigCloneBench dataset was employed for quantifying the presented technique. the results of the experiments depicted the supremacy of the presented technique over others and its F1 score was computed at 33.49% [4]. Bandi, et al. presented a review that focused

on formulating a device called Clone Swarm for detecting the clones in a project and effectively illustrating the information. This device was capable of mining any open-sourced GIT repository. GitHub was presented to provide the source code for the formulated device[5]. Xu, et al. conducted a study in which an enhanced SCCD-GAN was suggested to detect the semantic code clone based on a graph representation form of programs. This algorithm consisted of GAN (Graph Attention Network) for computing the similarity of code pairs. This algorithm offered a lower FPR (false positive rate) in contrast to the traditional techniques. Moreover, this algorithm offered higher precision[6]. Guo, et al. reviewed a complex network in the process of detecting software clones and a technique was projected based on this network to detect a clone code. This technique was adaptable for investigating similar sub-networks so that the clones were detected. The publicly available code was detected using the projected technique and its re-utilization was done in software to analyze the security[7]. Wang and Liu conducted a study in which a new ICCV (image-based clone code detection and visualization) method was introduced based on image processing. Initially, the comments, whitespace, etc. were eliminated to pre-process the source code. Subsequently, this method was implemented to transform the processed source code into pictures and normalize these images. Eventually, the information related to the clone code was detected and visualized using the Jaccard distance and perceptual hash algorithm. The experimental outcomes demonstrated the introduced method yielded an accuracy of 100% for detecting type-1, 88% for type-2, and 60% for type-3 clone code[8].

Bowman et al. emphasized developing a method recognized as VGRAPH to recognize vulnerable code clones. This method had robustness for modifying the code. Furthermore, a matching algorithm was put forward based on 3 graph-based elements which had the potential for detecting the code cloning, and the precision obtained from the developed method was counted at 98% and recall was 97%[9]. Othman and Kaya aimed to implement the technique of detecting the clone by recognizing the clone code and replacing it with a single call to the function. In this, the function was utilized to simulate the behavior of one instance of the clone group. The refactoring IDE was overviewed in this research. The process to detect the clone and diverse aspects of cloning were presented. The XML format was executed to generate the source[10]. Matsushima and Inoue conducted a study in which a technique was established to compare and visualize the results after detecting the outcomes on the clone pairs. This technique assisted the developers in contrasting the results and diverse metrics. The results of the comparative analysis revealed that the established technique performed well with the implementation of two tools: CCFinderX and NiCad[11].

Li, et al. recommended a mechanism based on a method based on EET (event embedding tree) and GAT (Graph Attention Network) for detecting the code clone. A program control flow graph was utilized to capture every statement's execution attributes and extract the context association of diverse statements in the control flow. The experimental outcomes proved that the recommended mechanism performed more effectively as compared to other techniques while detecting the clone of Type-3 and Type-4[12].

2 Research Methodology

This article aims to analyze various research which is done in the field of code clone detection. The code clone detection techniques are analyzed systemically by following certain processes. The search process follows various steps which are represented in Figure 1.

Fig 1. Research Method

The research method steps are described which were used for the systemic review. The first step is to define the research question based on which the search criteria will be finalized. When the search criteria will be defined in the next step search repository will be finalized. The popular search repositories are ACM, Springer, IEEE, and Science Direct. The data will be searched from the search repositories which will be followed by the inclusion and exclusion criteria. The inclusion and exclusion

criteria will be selected based on the years and techniques. The data will be included or excluded based on the search criteria. When the data get finalized then the data will be compiled for the conclusion.

2.1 Research Questions

The aim is to design research questions to review the papers based on certain criteria:

RQ 1: What percentage of the techniques designed so far are based on text-based, token-based, tree-based, and graph techniques?

RQ 2: What percentage of Metric Based techniques are available on different sources?

RQ 3: Do some hybrid techniques available for code clone detection?

2.2 Search Process

The search process describes various steps which are a source of information, search criteria, and study selection. Each step is described below in detail:

2.2.1 Source of Information

The source of information is from the various repositories which are available online like google scholar and Science Direct. Google Scholar provides various types of research articles that are available at various conferences and journals. Google Scholar contains various repositories like ACM Digital library (https://dl.acm.org), IEEE Explore (https://ieeexplore.ieee.org), Elsevier (https://www.elsevier.com), Springer (https://www.springer.com/in).

2.2.2 Search Criteria

The search criteria started from the search string which is code clone detection techniques from google scholar. The search results showed 2,56,000 available articles on Google scholar. The search criteria were narrowed down to the starting year as 2010 and the ending year as 2020. The total number of available articles is 66000 which are available on google scholar. The search strings were modified from code clone detection to code clone detection using text-based techniques. The research results showed 16400 articles that were published on google scholar. The next search string used was code clone detection using token-based techniques and the number of articles that are available on google scholar will be 18300. The 19200 articles are searched for google scholar when the search string was modified to code clone detection using tree-based techniques. The graph-based techniques show 16400 articles that are available on code clone detection. The metric-based techniques show impressive results of 19500 articles which are available on google scholar. Figure 2 shows the graphical representation of the available articles.

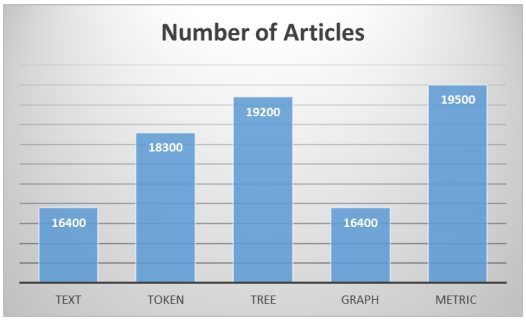

Fig 2. Number of Articles Available on Google Scholar

2.2.3 Comparative Analysis

The various code clone detection techniques are compared based on the various parameters. The code clone detection techniques are broadly classified into text-based, token Based, Graph based, and metric-based techniques. The techniques are compared in terms of accuracy, precision, recall, robustness, and scalability as shown in table 1. The parameters of the comparison are described below:

1. **Accuracy**: - The accuracy directly describes how accurately the clone will be detected from the code. The accuracy of the model will be increased when the false positive values get reduced at the time of detection

2. **Precision**: - The technique should be considered good when the precision value of the model is high. The model is good and the value of precision is increased when the false positive value is high.

3. **Recall:** - The recall value should be high when the clones are detected accurately. The recall value gets increased when the false positive values will be recalled from the model.

4. **Scalability**: - The model must be scalable and accurate. The model will be scalable when the model is tested over the large code it gave the same performance as when tested on small code

5. **Robustness**: - The code clone is of various types like type-1, type-2, etc. The model will be robust and can detect the maximum number of clone types.

Table 1. Comparative Analysis of Techniques

Technique Type	Accuracy	Precision	Recall	Scalability	Robustness
Text-Based	High	High	High	Less Scalable	High Robustness
Token-Based	Minimum	Moderate	Moderate	Scalable	Medium Robustness
Tree-Based	Minimum	Medium	Moderate	High Scalable	Low Robustness
Graph-Based	High	Medium	High	Scalable	High Robustness
Metric Based	High	High	High	High Scalable	Medium Robustness

2.2.4 Study Criteria

The study criteria are based on the articles which are available on google scholar. The research articles are searched on the type of techniques which are text-based, token-based, tree-based, graph-based, and metric-based techniques. The text-based techniques which are approx. 12 articles are selected for the review for code clone detection. A total of 8 articles are selected from the token-based techniques. The tree-based and graph-based techniques articles are 10 and 15 respectively selected for the analysis. The 20 research articles are selected which are based on metric-based techniques.

3 Result and Discussion

This study conducted a systematic review and analysis of code clone detection techniques. The systematic review is generated based on the research questions. The result of each question was deeply analyzed to draw some inferences regarding usage, application, research trends, future needs, and research directions.

3.1 What percentage of the techniques designed so far are based on text-based, token-based, tree-based, and graph techniques? (Research Question 1)

The papers which are downloaded from different sources are categorized according to techniques which are text-based, token-based, tree-based, and graph-based techniques. Text-based techniques are less available compared to token-based techniques. The tree and graph-based techniques are very popular for code clone detection. Approx. 20 percent of articles are available on text-based techniques for code clone detection. 30 percent of articles are available on token-based techniques for code clone detection. The rest 50 percent of the articles are available on the tree-based and graph-based techniques. The major part of the available articles is based on tree-graph-based techniques as shown in Figure 3.

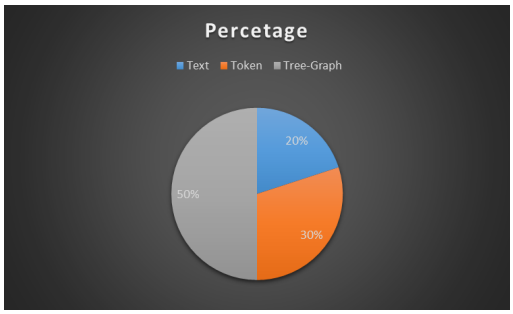

Fig 3. Percentage of Data Available

3.2 What percentage of Metric Based techniques are available on different sources? (Research Question 2)

The percentage of data sharing is shown in Figure 4 below. Books contribute only 8% of study and research material available on Metric Based code clone detection while conferences and journals share an approximately equal share of 46 percent each.

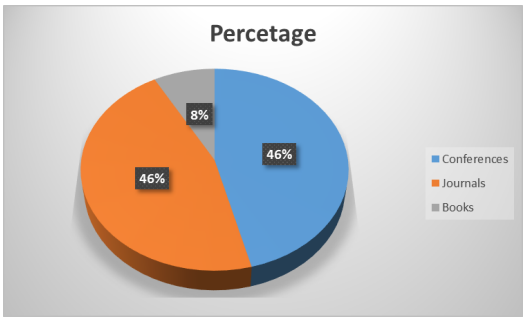

Fig 4. Percentage of Data on Metric-Based Techniques

Data Collection from various sources like Google Scholar, Science Direct, and other sources was conducted four times over 15 months and the percentage of papers from various sources was analyzed and has been represented in Figure 5 below.

	1/8/2021	1/12/2021	1/5/2022	1/10/2022
Google Scholar	80 %	65 %	60%	73%
Science Direct	12 %	25%	30%	12%
Others	8%	5%	10%	15%

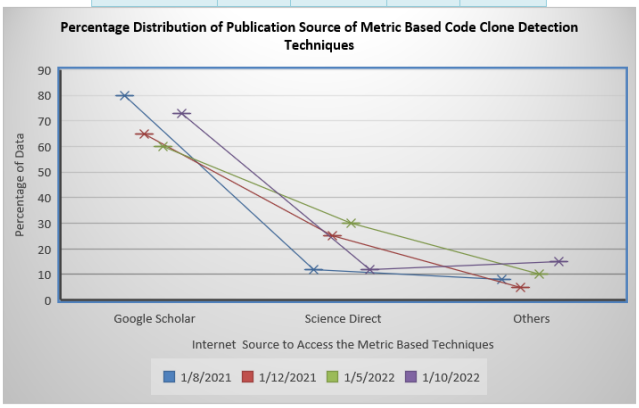

Fig 5. 5: Percentage Distribution of Publication Sources of Metric Based Clone Detection Techniques

3.3 Do some hybrid techniques are available for code clone detection? (Research Question 3)

Recently, many researchers are implementing new combinations of techniques for code clone detection. The hybrid techniques which are available for code clone detection are based on Neural networks, machine learning, and data mining-based techniques. Another major set of hybrid techniques is based on nature-inspired techniques for code clone detection. Many other approaches, such as image similarity, attention networks, attentive graph embedding, program slicing-based approaches, pairwise feature fusion, etc., were encountered during the search process. Various repositories were searched and analyzed to find the pattern and inclination of research development of different hybrid techniques for code clone detection as shown in Figure 6 below.

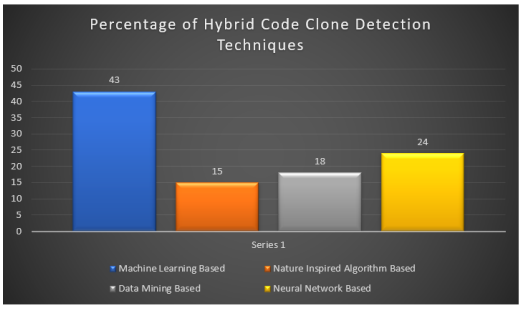

Fig 6. Percentage Distribution of Types of Hybrid Clone Detection Techniques

4 Conclusion

Code clones can be defined using diverse ways such as reusing code that the developers utilized the most. Such changes are considered to modify and enhance the performance of a software system, resulting in code cloning. The various types of techniques for code clone detection are compared in terms of accuracy, precision, recall, scalability, and robustness. The main findings related to research objective 1 depicted that 50% of total clone detection techniques are tree and graph-based Code Clone Detection techniques followed by 20% of text-based and 30% of token-based code-clone detection techniques. The second objective was achieved by exploring the fact that an equal percentage of 46% of research work related to metric-based code clone detection techniques has been published in journals and conferences. The rest of the 8% of metric-based clone detection techniques have been published in books. The findings related to the third objective exhibited that 43% of hybrid code clone detection techniques are based on machine learning techniques, 24% are based on neural networks, and 18% of techniques are data mining based followed by 15% nature inspired based algorithms. After analysis, it is analyzed that the metric-based technique is the most reliable technique for code clone detection. The main conclusion drawn lies in the fact that there is an urgent need to develop clone detection techniques that can detect all four types of clones together. Most of the available techniques can capture only a single type of code clone detection. In this paper, the scientific area of code cloning is examined from a macroscopic standpoint. The results and information in this study can be used as a reference for future research to further examine each sub-research field.

References

1) Shobha G, Rana A, Kansal V, Tanwar S. Code Clone Detection—A Systematic Review. In: Hassanien AE, Bhattacharyya S, Chakrabati S, Bhattacharya A, editors. Emerging Technologies in Data Mining and Information Security;vol. 1300. Springer. 2021;p. 645–655. Available from: https://doi.org/10.1007/978-981-33-4367-2_61.
2) Saini N, Singh S, Suman. Code Clones: Detection and Management. *Procedia Computer Science*. 2018;132:718–727. Available from: https://doi.org/10.1016/j.procs.2018.05.080.
3) Yuan D, Fang S, Zhang T, Xu Z, Luo X. Java Code Clone Detection by Exploiting Semantic and Syntax Information From Intermediate Code-Based Graph. *IEEE Transactions on Reliability*. 2022;p. 1–16. Available from: https://ieeexplore.ieee.org/document/9792461.
4) Bandi V, Roy CK, Gutwin C. Clone Swarm: A Cloud Based Code-Clone Analysis Tool. *2020 IEEE 14th International Workshop on Software Clones (IWSC)*. 2020;p. 52–56. Available from: https://ieeexplore.ieee.org/document/9047642.
5) Xu K, Liu Y. SCCD-GAN: An Enhanced Semantic Code Clone Detection Model Using GAN. *2021 IEEE 4th International Conference on Electronics and Communication Engineering (ICECE)*. 2021;p. 16–22. Available from: https://ieeexplore.ieee.org/document/9674552.
6) Guo H, Ai J, Shi T. A Clone Code Detection Method Based on Software Complex Network. *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 2019;p. 120–121. Available from: https://ieeexplore.ieee.org/document/8990341.
7) Wang Y, Liu D. Image-Based Clone Code Detection and Visualization. *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*. 2019;p. 168–175. Available from: https://ieeexplore.ieee.org/document/8950911.
8) Bowman B, Huang HH. VGRAPH: A Robust Vulnerable Code Clone Detection System Using Code Property Triplets. *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2020;p. 53–69. Available from: https://ieeexplore.ieee.org/abstract/document/9230372.
9) Othman ZS, Kaya M. Refactoring Code Clone Detection. *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*. 2019;p. 1–6. Available from: https://ieeexplore.ieee.org/document/8757479.
10) Matsushima K, Inoue K. Comparison and Visualization of Code Clone Detection Results. *2020 IEEE 14th International Workshop on Software Clones (IWSC)*. 2020;p. 45–51. Available from: https://ieeexplore.ieee.org/document/9047633.
11) Li B, Ye C, Guan S, Zhou H. Semantic Code Clone Detection Via Event Embedding Tree and GAT Network. *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. 2020;p. 382–393. Available from: https://ieeexplore.ieee.org/document/9282778.
12) Kumar A, Yadav R, Kumar K. A Systematic Review of Semantic Clone Detection Techniques in Software Systems. *IOP Conference Series: Materials Science and Engineering*. 2021;1022(012074). Available from: https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012074.

13) Elkhail AA, Svacina J, Cerny T. Intelligent token-based code clone detection system for large scale source code. In: Proceedings of the Conference on Research in Adaptive and Convergent Systems. ACM. 2019;p. 256–260. Available from: https://doi.org/10.1145/3338840.3355654.
14) Khazaal YM, Hammo AY. Survey on Software code clone detection. *Technium: Romanian Journal of Applied Sciences and Technology*. 2022;4(3):28–36. Available from: https://doi.org/10.47577/technium.v4i3.636.
15) Tronicek Z. Indexing source code and clone detection. *Information and Software Technology*. 2022. Available from: https://doi.org/10.1016/j.infsof.2021.106805.
16) Jo YB, Lee J, Yoo CJ. Two-Pass Technique for Clone Detection and Type Classification Using Tree-Based Convolution Neural Network. *Applied Sciences*. 2021;11(14):6613. Available from: https://doi.org/10.3390/app11146613.
17) Runwal AN, Waghmare AD. Code Clone Detection based on Logical Similarity: A Review. *IJSRSET*. 2017;5:148–151. Available from: https://ijarcce.com/upload/2017/september-17/IJARCCE%208.pdf.
18) Khazaal YM, Hammo AY. Survey on Software code clone detection. *Technium: Romanian Journal of Applied Sciences and Technology*. 1920;4(3):28–36. Available from: https://techniumscience.com/index.php/technium/article/view/6361.
19) Ain QU, Butt WH, Anwar MW, Azam F, Maqbool B. A Systematic Review on Code Clone Detection. *IEEE Access*. 2019;7:86121–86144. Available from: https://ieeexplore.ieee.org/document/8719895.
20) Zhang X, Wang T, Yu Y, Zhang Y, Zhong Y, Wang H. The Development and Prospect of Code Clone. *Arxiv*. 2022. Available from: https://arxiv.org/pdf/2202.08497.pdf.