EasyChair Preprint

№ 6641

Detection of Metrics Based Code Cloning Using Optimised SVM Algorithm

S Karthik and B Rajdeepa

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

September 21, 2021

Detection of Metrics based code cloning using Optimised SVM Algorithm

[1] S.Karthik, Research Scholar & Assistant Professor
Department of Information Technology, PSG College of Arts & Science, Coimbatore-14.
karthikyessekar@gmail.com
[2] Dr. B. Rajdeepa, Associate Professor & Head
Department of Information Technology, PSG College of Arts & Science, Coimbatore-14.
rajdeepab@gmail.com

Abstract

Code segments usually arises due to replication from one place and then rewrite them in to another segment of code with or without variations / changes are software cloning and the copied code is called clone. Generally Code Cloning defines the designing and development of software systems. Detection can be done based on Textual analysis, Lexical analysis, Syntax analysis, Se-mantic analysis, Hybrid analysis and Metric analysis. The major drawback of the research is that it emphasises more on fragments of copied code and does not focus on the feature that the fragments of duplicated code are may be part of a larger replicated program. In this process, many methods take a lot of time and it generates intricacy. In our study, a program source code is inspected for detecting various methods by adopting an "OPTIMIZED SVM ALGORITHM" and the method definitions are extracted and collected by means of a CLONE CODE. To evaluate the performance parameters we calculate the LOC, the number of recurrences, and maximum and minimum LOC. To enhance the performance metrics precision recall, accuracy and reduce the error rate and time complexity.

Keywords: *Clone Detection; SVM; Functional Clones; Fitness Function.*

1. Introduction

Software program development and fortification will be the most current and also big-budgeted time of SDLC [1]. The major difficulty behind the product assist is the exchange of modern programming framework by using new functionalities, to rectify errors within the product framework or due to the new necessities of the cooperation that would not recognized among those prerequisite phase. However, the maximum severe endeavours are required while extending the modern programming by way of such as new functionalities. One of the frameworks used for programming assist is software re-engineering is the maximum utilized [2]. Software program Re-engineering is to study the modern framework and manufacture it once more with higher functionalities.

In software re-engineering code cloning is done by way of reusing code as it is or differently with a few changes. This technique is referred to as code cloning. It is observed to be a major issue in industrial framework.

In the existence of clones, the everyday functionalities of the sys-tem might not be selected. However, without control measures with aid of the maintenance group, similarly enhancement as well become prohibitively costly.

1.1. Definition of code clone

Replicating code sections and afterward reuse by sticking with or without minor changes or variations are commonplace activities in programming advancements. This kind of reuse approach of existing code is called code cloning and the stuck code fragment (with or without alterations) is referred to as the image of the unique [3], [4]. Replicating square are referred to as clones and copying execution, including little modifications is cloning. Code cloning is the major issue in industrial framework. Moreover, it also increase the redundancy in the program and it also frequently suggest the layout issues [5]. Figure 1 shows code with clone.

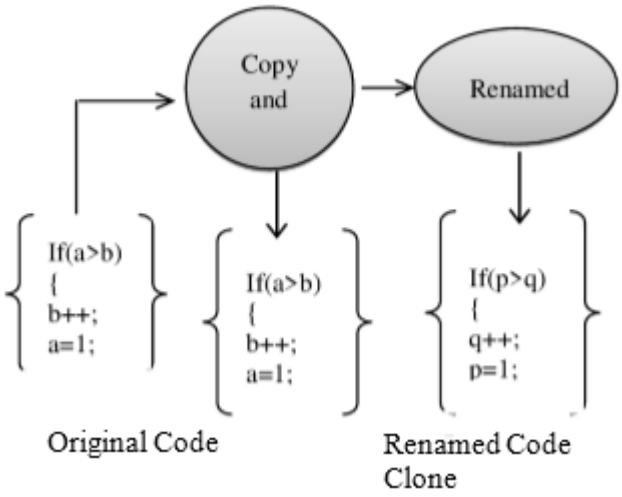

Fig: 1.1 Original Code with Renamed Code

1.2. Types of code clone

Code clone possibly will be of any type that depend upon method of the developer and the capability of utilizing the code which differ from replicating as it is to duplicate the code, however with less change which would be done at diverse level in the method [1].
There are two types of similarities between two codes textual similarity or functional similarity. The clone belongs to first type is mostly the output of copy paste technique. The first three types come under the texture similarity and the fourth TYPE is under the functional clone [6]. Here we describe replica kinds relied at the type of indistinguishable paired code sections could be:
Textual Similarity: A binary code fragments may be related to the similarity in their source code text. The varieties of clones are described in an effort to seek texture similarity.
TYPE I: It is known as exact clone in which a copied code component is identical or same to the specific code fragment besides for a few feasible variations in comments and whitespaces.

TYPE II: A Type II clone is same as a TYPE I clone. In this there exist fragments which are syntactically identical except variation in comments, whitespaces, literals, identifiers, tokens and so on.
TYPE II is also referred to as renamed clones.

TYPE III: In this type copied the fragments with further modification consisting of changed, added or removed statements at the side of various dissimilarities in format, identifiers, comments, literals, and types. The agency of code fragment can be altered and they will even look or act slight in a different way. This type of clone is hard to be located, for the purpose that the wholly framework expertise is needed. It is also known as Gapped clones.
Functional Similarities: Similarity factor of two code fragments are often based on their functionalities. If the two code fragments has comparable functionalities, then these are referred as TYPE IV clones or semantic clones [7].
TYPE IV: These type of clones are also referred to as semantic or logic clones. It provides an alternate approach to resolve the same issues. In this category of specific clones, the cloned part ís not necessarily copied from the first one. Two code fragments may possibly be established by two different programmers too.

Code clone detection techniques

Software program clone detection is generally carried out for the rearrangement, maintenance, refactoring and reengineering of software. Clone detection process, minimized the maintenance cost and reduce redundancy in the program and make that soft-ware more efficient.
Software clone detection techniques are as follow:

1) Textual based Approach: It is the textual content based method where in the fragments of code are compared wíth every other to discover similarities of texts or strings. If similarity factor is true then code is cloned code [8], [9].
2) Token Based Approach: To somehow it is same as text based approach but instead of taking line of code directly for representation, it converts each and every line of code in to tokens. Since of tokenization phase this approach is slower than text based method. This approach is also referred as lexícal approach [10], [11], [12].
3) Syntactic Approaches: This technique also referred to as Abstract Syntax Tree (AST). In this method it uses a parser to convert source code in to AST which can be used to detect cloned code by using tree matching or structural metric method [13], [14].
4) Semantic Approaches: Some methods uses static application evaluation to provide specific facts than truly syntactic similarity. In some of the methods, the source code is represented as a PDG (Program dependency graph) [11].
5) Metric based Approach: For comparing code fragments directly, different metrics are computed and then compared to detect the clones from the code. This metric approach is more accurate and scalable and give more precise result as compare to other techniques [15].
6) Hybrid Approach: This approach is the combination of more than one technique to detect the cloned code. In this approach source code is represented syntactically and semantically both.

3. Related work

Since 1990's, Code clone detection is dynamic research locale. Detection of cloned clone is directly related to maintenance cost, reuse, code redesigning and making the code more reliable and efficient. The survey on the clone detection shows, that there are various different techniques and algorithms which are used for detection of clones.
E. Kodhai, et al presented a hybrid approach for detection of clones. In this research, hybrid approach is used to locate clones in multiple alterations of code. Metrics computation and textual analysis techniques are combined to design hybrid approach. The features of clone manager tool are enhanced in this research by integrating the code modification and detection functionalities in it. Results are evaluated on 6 projects which are publically available. The performance parameters used in this study are precision and recall [16]. Shruti et al. discussed about the importance of detection of similar clones. According to the author, detection of similar clones is equally important as the detection of identical clones. In this research, support vector machines are used. C language pro-grams are used to perform the detection of similar clones. Feature sets are generated using parser and for classification purpose, SVM is used. The code clone detection process can be performed by producing the feature sets for fragments of code and then matching process id done for finding the similarity. Libsvm and Tkinter tools are used in this research. Results are evaluated in the terms of accuracy [17]. Nils Gode et al. discussed about the algorithms used for detection of clones. In this study, incremental clone detection algorithm is proposed which can be used to find the clones in the number of versions of project or program. Previous version results are used to find the clones in the next version of the program. The tool named as iclone, is used in this study. The results of this research work are compared with the traditional detection techniques. 5 projects, each having 2 versions are used in this study. The results of this study shows that iclone provides best results as compare to traditional clone detection approaches [18].
Sushma et al. presented a technique for detection of module level code clones in projects. In this research, process of detection of code clones is performed with the use of software metrics. JS clone detection tool (JSCCD) is used to find the first and second type of clones. Java programs are used as input files for detection of code clones. Nine software metrics are used in this study. The precision and recall parameters are calculated in order to evaluate the results [19]. Yoshiki et al. presented code clone detection technique based on the PDG approach. In this paper, PDG based detection and specialization heuristics are proposed as the enhancement for PDG approach. Four open source projects are used in this study. Results of this study describes the effectiveness of this approach. Detection time is reduced in this research work [20].
Kavitha et al. implemented a novel technique for code clone detection. In this study, FP-growth technique is used to detect simple and functional code clones. According to the author FP-growth method is suitable for only type 1 and type four clones. The results describes that data mining techniques

are best suitable for detection of clones [21]. Yang Yuan et al, presented an accurate and efficient token based technique. To define feature matrices, Boreas define a new counting approach. It was proficient to consider the accuracy and minimize the compilation time by using tool Deckard [22].
Mohammed Abdul Bari et al, defines with antiquity and contextual concept of code cloning, describes over-all classification of recent approaches and tools, classify development tools in binary dissimilar format as static-code clone and vibrant code cloning, both accessible with the executable program, as a solution, the static-code is four types; Type-1, Type-2, Type-3 and Type-4, to completely design a procedure to perceive and eliminate code cloning [23].

4. Proposed approach

Previous studies prove that machine-learning algorithms are suitable for incremental clone detection process and wide in industrial practice [23], [24], [25]. In our research work, optimized code manager based on support vector machine is used. Optimized code manager with input parameters are used in present study. Optimized support vector machine use supervised machine learning algorithm for training purpose. Figure 2 shows proposed flow chart.

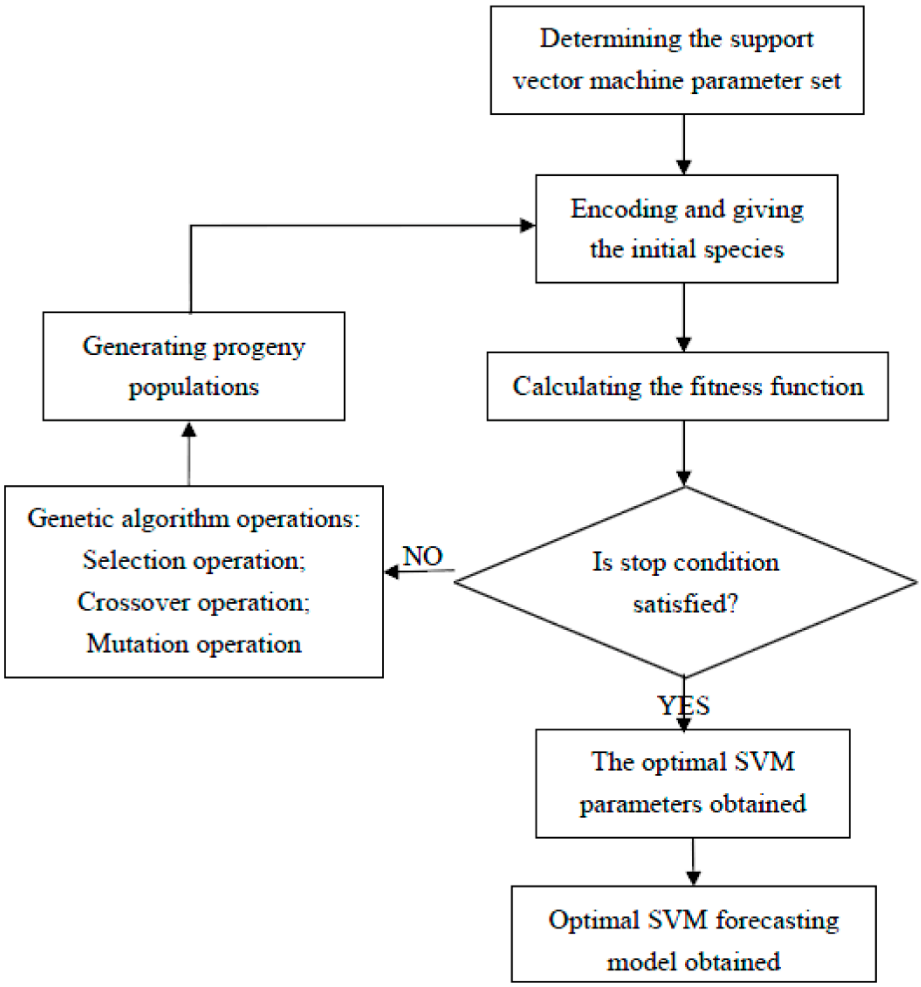

Fig:2 Proposed Flow Chart for Optimized support vector machine use supervised machine learning algorithm

a) Artificial Bee Colony Optimization (ABC)

In this algorithm, an artificial bee colony consists of working bees, onlookers and scouts. It contains bystander bee i.e; bee coming up on the dance area to obtaín the information about food sources, and employed bee i.e; a bee going to the food source, and a guide bee i.e; a bee carrying out accidental

search [31]. ABC is used to filter the data. In this, cost fitness function is used to Re-filter the data, which gives the most favourable solution for the desired problem. Figure 3 shows fitness function

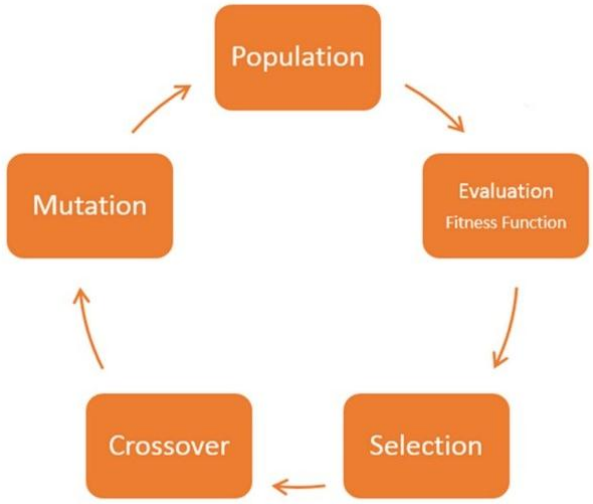

Figure 3: Fitness Function

b) Optimized Support Vector Machine (Code Clone Optimized Manger)

An optimization technique is designed to resolve various difficulties. This technique is used to initialize the population i.e; set of size. It is used to provide the new population that would be superior to other previous one. The fitness function is used to select the results which are particular to form new solution. The optimization techniques can solve an optimization issue by using the following three operators: (1) Selectíon (2) Crossover and (3) Mutatíon [26].

Optimized Support Vector Machines have been expansively re-searched in the information mining and learning populations for the last time and dynamically applied to requests in several fields. Optimized Vectors are normally used for machine learning classification [27], ranking methods for which they are known as classifying optimized support vector machine. Optimized Support Vector Machine is two special properties are that OSVM achieve:

(i) High-generalization by maximizing the margin and

(ii) Support an effective learning of non-linear methods by kernel trick.

Optimized SVM helps to perform the mapping between the input-space and feature-space to support non-linear classification issues [3], [18]. The kernel function is harmful for responsibility this by permitting the absence of the exact formulation of the method of mapping which can cause the issue of curse of dimensions. This makes a linear classification in the novel-space equal to non-linear classification in the input-space. Based on the need of extending the knowledge in the field of detection of clones, the proposed method emphasize on the detectíon of clones using an optimized code manager which is the combination of textual analysis and metrics computation to calculate the performance parameters. In Optimized SVM there are two section training section and testing phase in which code files are trained and tested. In training Phase, there is pre-processing of the uploaded code files and extract the features which is known as metric computation. In SVM [28] the classification is done by hyper-plane (kernel function) [29] [30] in this study Radius bias function (Rbf) is used for linear and nonlinear classification because Rbf is string based.

5. Experimental evaluation

Here we present the empirical analysis of our proposed technique. For implementation, we used a tool named MATLAB. To start with, the produced datasets are defined in section 5.1. Second, the experimental outcomes of our proposed procedure in section 5.2. Lastly, the analysis of our progressive tool for precision, recall, and accuracy are depicted in section 5.3.

5.1. Experimental setup

Various datasets are used to find the clones and system's efficiency. Here in table 1 the data sets like wget, struts etc. used with various revisions. It consists of 1 project from C language and 4 from JAVA

with significantly different sizes. Multiple revisions of the program are available, which makes these programs reliable in use each program has a different size.

Table 1: Datasets for Code Clone Detection

SNo	Project Name	No. of Revisions	Max Files	MinLOC	MaxLOC
1	Wget	10	114	3657	22294
2	Struts	11	1928	107723	270734
3	jhotdraw	8	680	17960	135400
4	Jfreechart	6	1007	282368	332546
5	jedit	8	549	170098	177768

5.2. Results and discussions

In this section the changes to the mentioned projects in table 1 are measured for multi clone detection process on each revision. The table 2 shows the number of clones detected for the dataset by using OSVM. Detection time is one of the important factor when working with the various large datasets. In table 3, it explained the experimental results programs from C language have been selected, which be different significantly size.

Table 2: Detected Clones on Uploaded Datasets

Revision	Wget		Strut		jHotdraw		Jfreechart		Jedit	
	ICM	OSVM	ICM	OSVM	ICM	OSVM	ICM	OSVM	ICM	OSVM
1	14	19	11980	13697	1018	1192	129800	135962	9843	10934
2	15	21	12113	12654	1103	1262	130741	130965	9843	9920
3	15	17	12832	12953	1902	2182	130796	134667	9849	9967
4	15	17	13906	14268	1919	2045	130819	132775	9855	9826
5	15	17	13911	14698	2091	2186	130997	132698	9736	9833
6	17	23	13965	14256	2524	2658	131234	131954	9738	9826
7	37	48	14078	14965	2605	2785	-	-	9848	9965
8	37	43	14078	14968	2697	2675	-	-	9891	9936
9	37	43	14079	14962	-	-	-	-	-	-
10	41	47	14244	14987	-	-	-	-	-	-
11	-	-	14287	14985	-	-	-	-	-	-

Table 3: Time Comparison for Detection

Revision	Wget		Strut		jHotdraw		Jfreechart		Jedit	
	ICM	OSVM	ICM	OSVM	ICM	OSVM	ICM	OSVM	ICM	OSVM
1	5	3	1055	945	18	13	2545	2064	1268	1136
2	2	0.5	95	72	5	3	120	106	92	78
3	0.9	0.5	98	76	7	6	76	62	94	79
4	0.9	0.3	125	106	8	5	78	63	93	75
5	0.9	0.3	45	43	12	9	95	78	96	79
6	5	2	54	29	13	10	110	91	92	78
7	3	1.9	65	35	16	14	-	-	95	76
8	3	1.9	68	38	17	14	-	-	112	92
9	3	1.9	64	35	-	-	-	-	-	-
10	3	1.9	86	54	-	-	-	-	-	-
11	-	-	88	58	-	-	-	-	-	-

5.3. Evaluation of parameters

Results of this study which is done to evaluate performance parameters such as precision, recall, and accuracy by using Optimized Code Manager. In this study calculated average precision of dataset is

0.98 approx whereas recall is approximately 0.99 and accuracy is recorded to be 97.02. Comparison of our proposed approach with existing one showed that Optimized code manager with 5 projects used as dataset provide better results than previous incremental clone detection approach.
Precision is referred to as positive predicted value is independent of accuracy as it can be precise, but can be inaccurate [32], [33].

Precision $= TruePositve\ /\ TruePositive + FalsePositive$
Recall is defined as a part of relevant data that have been fetched over the total set of relevant data [33].
Recall $= TruePositive\ /\ TruePositive + FalseNegative$

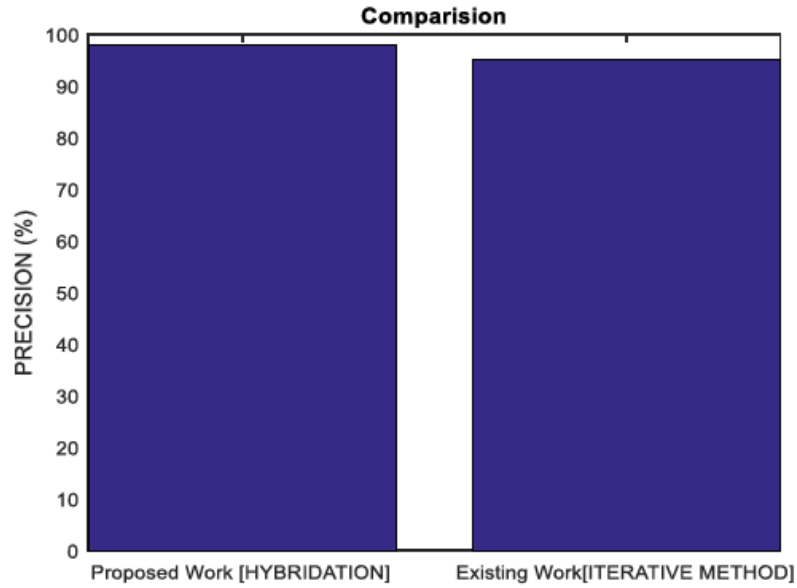

Fig 5: Precision Comparison

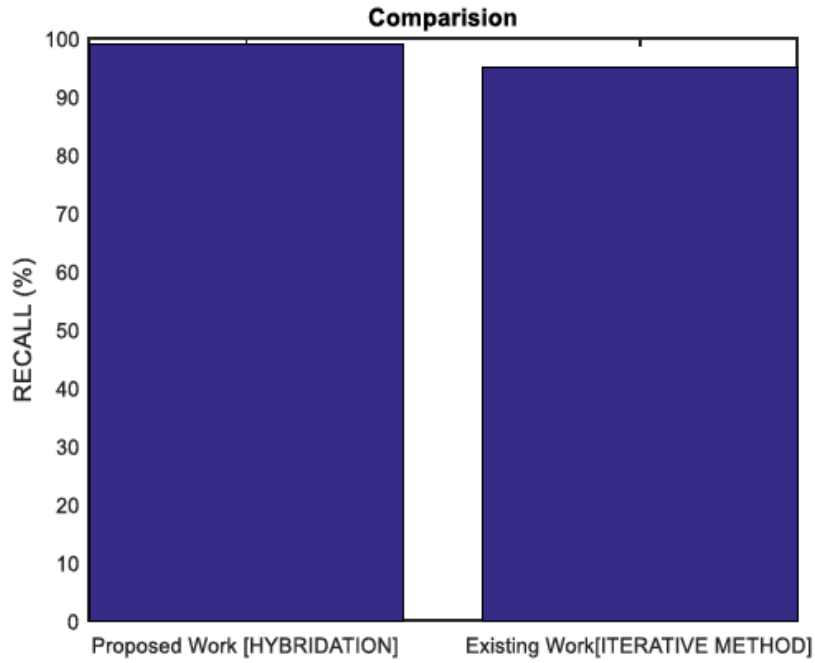

Fig. 6: Recall Comparison

Accuracy is refer to the proximity of a calculated value to the benchmark value

Accuracy= (TP + TN) / (TP + FP + TN + FN)

Where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

6. Conclusion

Our proposed approach used in this study provide optimization in clone detection process. Clone detection process requires high accuracy and precision. We compared the results with existing technique called incremental clone detection and observed that better results are obtained with OSVM.

After winding up the results, we conclude that this technique illustrate high precision, recall and accuracy and also it is low in rejected values.

References

[1] C.K. Roy, J.R. Cordy, A Survey on Software Clone Detection Re-search, Technical Report 2007-541, Queen's University at Kingston Ontario, Canada, 2007, p. 115.

[2] G. Kaur, "The Survey of the Code Clone Detection Techniques and Process with Types (I, II, III and IV)", *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 2, pp. 392-399, 2018.

[3] A. Sheneamer and J. Kalita, "Semantic Clone Detection Using Ma-chine Learning," *2016 15th IEEE International Conference on Ma-chine Learning and Applications (ICMLA)*, 2016.

[4] B. A. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE- 2007-01, School of Computer Science and Mathematics, KeeleUniversity, Keele and Department of Computer Science, University of Durham, Durham, UK, 2007, p.65.

[5] T. Kamiya, S. Kusumoto, K. Inoue, CCFinder: A multi linguistic token based code clone detection system for Large-scale source code, IEEE Transactions on Software Engineering 28 (7) (2002) 654–670.

[6] Bellon S et.al. Comparison and evaluation of clone detection tools. IEEE Transactions on Software Engineering vol 33(9) (2007), pp.577- 591.

[7] "Software clone detection: A systematic review"Dhavleesh Rattan, Rajesh Bhatia, Maninder Singh. Information and Software Technology, Volume 55, Issue 7, July 2013, Pages 1165–1199.

[8] S. Ducasse, M. Rieger, S. Demeyer, A language independent approach for detecting duplicated code, in: Proceedings of the 15th International Conference on Software Maintenance (ICSM'99), Ox-ford, England, UK, 1999, pp. 109–119.

[9] G. M. K. Selim, K. C. Foo and Y. Zou. "Enhancing Source-Based Clone Detection Using Intermediate Representation", Proc. WCRE, 2010, pp.227-236.

[10] S. Lee, I. Jeong, SDD: High performance code clone detection sys-tem for large scale source code, in: Proceedings of the Object Oriented Programming Systems Languages and Applications Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA Companion '05), San Diego, CA, USA, 2005, pp.140– 141.

[11] Roy CK, Cordy JR, Koschke R. Comparison and evaluation of clone detection techniques and tools: A qualitative approach. Science of Computer Programming 2009; 74:470–495.

[12] Patenaude, J-F, Ettore Merlo, Michel Dagenais, & Bruno Laguë, (1999) "Extending Software Quality Assessment Techniques to Java Systems", 7th International Workshop on Program Comprehension (IWPC'99), pp 49-56

[13] K. Kontogiannis, R. Demori, E. Merlo, M. Galler, M. Bernstein, Pattern matching for clone and concept detection, Automated Soft-ware Engineering 3 (1–2) (1996).pp.77–108.

[14] Roy CK, Cordy JR. A survey on software clone detection research. TR2007-541, Queen's School of Computing, 2007; 115.

[15] K. Kontogiannis, R. Demori, E. Merlo, M. Galler, M. Bernstein, Pattern matching for clone and concept detection, Automated Soft-ware Engineering 3 (1–2) (1996).pp.77–108.

[16] Kodhai, E., &Kanmani, S. (2016). Method-level incremental code clone detection using hybrid approach. International Journal of Computer Applications in Technology, 54(4), 279-289.

[17] Sn. Göde and R. Koschke, "Incremental Clone Detection", 2009 13th European Conference on Software Maintenance and Reengineering, 2009.

[18] Jadon, S. (2016, April). Code clones detection using machine-learning technique: Support vector machine. In Computing, Communication and Automation (ICCCA), 2016 International Conference on (pp. 399-303).IEEE.

[19] A Novel Metrics Based Technique for Code Clone Detection", International Journal of Engineering and Computer Science, 2016.
[20] Y. Higo and S. Kusumoto, "Code Clone Detection on Specialized PDGs with Heuristics", 2011 15th European Conference on Software Maintenance and Reengineering, 2011.
[21] K. Rajakumari and T. Jebarajan, "A novel approach to effective detection and analysis of code clones", Third International Conference on Innovative Computing Technology (INTECH 2013), 2013.
[22] Yuan, Yang, and Yao Guo. "Boreas: an accurate and scalable to-ken-based approach to code clone detection." In Automated Soft-ware Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on, pp. 286-289. IEEE, 2012.
[23] Yu, D., Wang, J., Wu, Q., Yang, J., Wang, J., Yang, W., & Yan, W. (2017, July). Detecting Java Code Clones with Multi-granularities Based on Byte code. In Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual (Vol. 1, pp. 317-326). IEEE.
[24] Dang, Y., Zhang, D., Ge, S., Huang, R., Chu, C., &Xie, T. (2017, May). Transferring code-clone detection and analysis to practice. In Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track (pp. 53-62). IEEE Press.
[25] Sheneamer, Abdullah, and JugalKalita. "Code clone detection using coarse and fine-grained hybrid approaches." In Intelligent Computing and Information Systems (ICICIS), 2015 IEEE Seventh International Conference on, pp.472- 480. IEEE, 2015.
[26] Fradkin, D., &Muchnik, I. (2006). Support vector machines for classification. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 70, 13-20.
[27] B. Joshi, P. Budhathoki, W. Woon and D. Svetinovic, "Software Clone Detection Using Clustering Approach", Neural Information Processing, pp. 520-527, 2015.
[28] Ju C, Guo F (2010) a distributed data-mining model based on support vector machines DSVM [J]. 30(10):1855–1863.
[29] Fuguang Wang, Ketai He, Ying Liu, Li Li and Xiaoguang Hu, "Re-search on the selection of kernel function in SVM based facial expression recognition", 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), 2013.
[30] Zhu S, Zhang R (2008) Research for selection of kernel function used in support vector machine [J]. SciTechnolEng 8(16):4513–4517.
[31] Karaboga, Dervis, and BahriyeBasturk. "On the performance of artificial bee colony (ABC) algorithm." Applied soft computing eight, no. 1 (2008), pp. 687-697.
[32] A. Beger, "Precision-Recall Curves", SSRN Electronic Journal, 2016.
[33] M. Buckland and F. Gey, "The relationship between Recall and Precision", Journal of the American Society for Information Science, vol. 45, no. 1, pp. 12-19, 1994.