Volume: 8 Issue: VI Month of publication: June 2020
DOI: http://doi.org/10.22214/ijraset.2020.6294

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

A Novel Approach to Detect Code Clone

Dr. R. V. Patil[1], Alfiya Pathan[2], Pooja Devkar[3], Ankita Charahate[4], Pragati Rajure[5]
[1]H.O.D. of Computer Department, Computer Engineering PDEA COEM.
[2, 3, 4, 5]Student, Computer Department, Computer Engineering PDEA COEM, India

Abstract: In software development process, coping of existing code fragment and pasting them with or without modification may be a frequent process.
Clone means copy of an ingenious form or duplicate. Software clone detection is vital to scale back the software maintenance cost and to acknowledge the software during a better way.
There are many software code clone detection techniques like text based, token-based, Abstract Syntax tree based etc. and they are used to spot and finding the existence of clones in software system. One of the approach is token based comparisons of two programs to detect code clone.
Our technique uses the token-based approach using hashing algorithm by analysis of two source codes in the process of finding clones among them and the percentage of cloning is reported as result.
Keyword: Code clone, lex, Tokenization, Detection technique, token based approach using hashing algorithm.

I. INTRODUCTION

Some of the software developers makes a replica of pieces of code and paste without making any changes like naming , updating or removing etc. to eliminate time and efforts. When code with a software vulnerability is copied, the vulnerability may continue to exist in the copied code if the developer is not aware of such copies.[1] Refactoring duplicate code can improve many software metrics, such as lines of code, cyclomatic complexity, and coupling. This sort of work doing the method of copying and adding the request code within the software development is called as code Cloning. Reusability has become common in recent software development by the developers, but it has various drawback . It leads to increase in maintenance cost which also decreases quality of the software in a system . As the code is copied without alternations , there is a chance of increase in bugs in the software, because errors present in one module can be increase the errors .

We all know that growing software becomes heavy day by day with age and high change requirement. The size of software increasing with growing age will lead to memory and performance related issues with existing infrastructure. This also raises the cost of maintenance. All these issues are related to re-engineering of software, studied under ambit of reverse engineering, branch of software engineering.

Reverse engineering is a broad research field, which starts from code and results to different artifacts that's gives information of design, and architecture of software.

Re-engineering of software can be done using software refactoring which includes various refactoring metrics and code clone detection. In this paper, we are discussing about code clone detection. The code clone detection will detect the amount of clones available in project or application.

This will also reveal the design related flaws or gap between design and implemented code. This will also state the amount of code can be refractor based on object oriented design flaws.

We are making application such that which will compare two files to find code clone from source file from target file .To check code clones you have to give source file and target file then click submit button then application will detect the code clone and will state type of code clone as type 1,type 2 etc.

There are many methods to detect code clones as textual detection method , token based method syntax method and syntactical method and combination of some methods

But we are using token based method to detect the code clone between two files. The token based approaches are also called lexical approach. In this technique, the whole source code is divided in to tokens by lexical analysis and then all the tokens are formed in to a set of token sequence.

normalize the code in form of token. So every line of source code is transformed into tokens then comparison applied on intermediate representation of code. The sequences of lines are compared through different algorithms. This technique is slower than text based approach and it is more robust.

II. METHODOLOGY

A. Clone Detecting Process

Clone Detecting is a process in which the input is source files and the output is clone-pairs. The entire process of Our token-based clone detecting technique consists of four steps:

1) *Lexical Analysis:* Each line of source files is divided into tokens corresponding to a lexical rule of the programming language. The tokens of all source files are concatenated into a single token sequence, so that finding clones in multiple files is performed in the same way as single file analysis. At this step, the white spaces between tokens are removed from the token sequence but the spaces are sent to the formatting step to reconstruct the original source files.
2) *Transformation:* The token sequence is transformed by sub-processes (2-1) and (2-2) described below. At the same time, the mapping information from the transformed token sequence into the original token sequence is stored for the later formatting step(2-1)Transformation by the transformation rules. The token sequence is transformed i.e. , tokens are added removed or changed based on the transformation rules.(2-2)Parameter replacement After step (2-1) each identifier related to types, variables, and constants is replaced with the special token (this replacement is a pre-process of the parameterized match proposed in [1]). This replacement makes code-portions in which variables are renamed to be equivalent token sequences.
3) *Detection:* From all the substrings on the transformed token sequence equivalent pairs are detected as a quadruplet (cp,cl,op,ol), Where cp and op are respectively, the position of the first and second portion , and cl and ol are their respective lengths.
4) *Formatting:* Each location of clone-pair is converted into line numbers on the original source files.

III. FIGURE

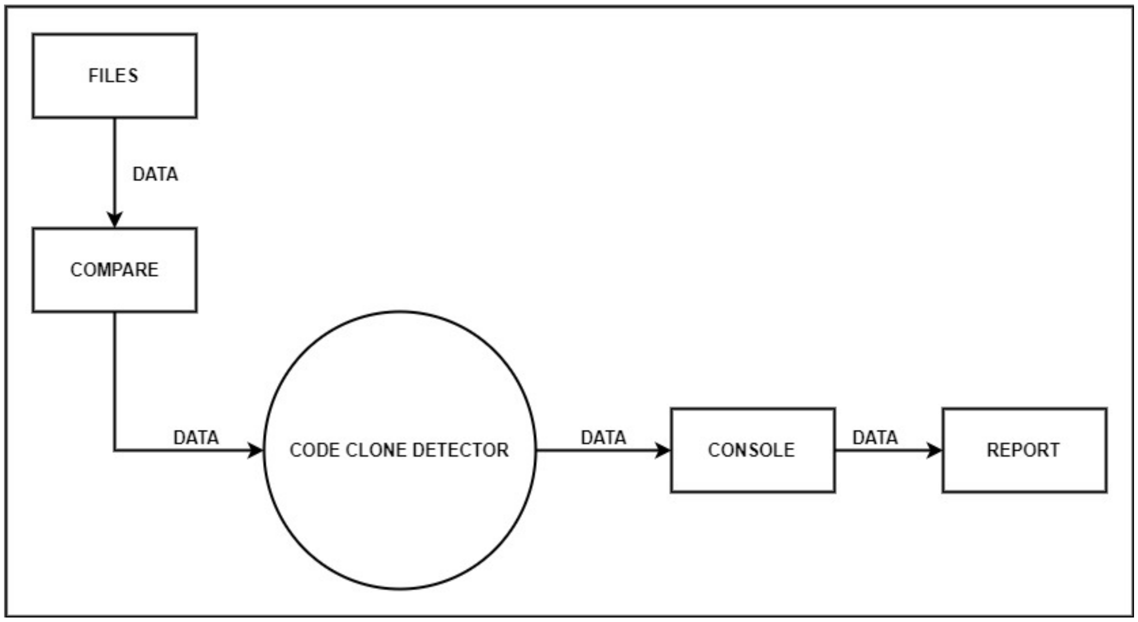

Fig. 1 Code Clone Detection Process

IV. SYSTEM METHODOLOGY STEP BY STEP

The working methodology of the code clone detection system is as given below:

A. The System is developed using various Python Libraries such as the filecmp, re, os etc.
B. The Graphical User Interface of the system is created using PySimpleGUI for Python.
C. The System has two screens. First Screen has the name of the system with a loading bar or a progress bar. Once the system is initialised properly the second screen is opened.
D. This is the main screen of the system. The main screen has two file picker used for picking the code files that needs to be compared.
E. After the user selects the files that needs to be compared the user can press the Submit Button.
F. Following actions takes place when the user presses the submit button:

1) Two temporary blank files are created.
2) The contents of both the files submitted for comparing are written in two new temporary files.
3) A function which is specially defined to remove white spaces in the file starts peforming its work and removes all the unnecessary white spaces.
4) A regex defined to remove comments in the code also starts performing its actions and removes the comments in the system.
5) Once both these operations are performed the system saves the temporary files and the compares it using filecmp inbuilt function. If the files are found to be similar then it stores True in a intermediate step variable and if it is dissimilar it stores False.
6) This intermediate variable would be used at the end.
7) Now again the files that needs to be compared is taken into two new temporary files.
8) The operation performed earlier are again performed but this time two more functions are performed. These two functions are again new regex fuctions for finding variables in the system.
9) The variables determined are compared and another intermediate result is stored in True (for similar) or False(for dissimilar) terms.
10) Now a logic of if else is used. If the intermediate result 1 is True the output is : Type 1

If the intermediate result 1 is False and intermediate result 2 is True the output is : Type 2.

If both results are false then output is : Cannot Identify.

11) Once the output is derived the temporary files generated during the process are deleted to save space and cleanup garbage.
12) The output is displayed using a Popup Window.

V. LITERATURE SURVEY

Sr.no	Paper Name	Author Name	Conclusion
1	A Approach for Detecting Type-IV Clones in Test	CoNovelde Brent van Bladel, Flanders Make vzw Belgium	It detect type 4 as well as type 3,2,1
2	CloneTM: A Code Clone Detection Tool Based on Latent Dirichlet Allocation	Sandeep Reddivari Mohammed Salman Khan	Support a variety of programming languages and adopt different clone detection strategies at different levels of complexity
3	Detecting Java Code Clones Based on Bytecode Sequence Alignment	Dongjin Yu , (Member, Ieee), Jiazha Yang, Xin Chen, And Jie Chen	It detects data clone of type 3,2,1.
4	A Systematic Review on Code Clone Detection	Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam, Bilal Makbul	It gives information on all different types of data clone types,and its tools.

VI. CONCLUSION

In this paper, We presented a clone detecting technique with transformation rules and a token-based comparison. They were applied to two files of codes in the experiments. An Experiment to compare two files found several subsystems that would come from a same original. Some of them have distinct code, and some are duplicated with in a files.

The existence of code clones in a program enhancement is conservation cost as their existence makes the execution program complex and generates the issue of redundancy. The study of prior research work suggests the major focus of their research work on implementation approaches for detection of identified clones. In the current research study, the main focus is on the development of a noble approach to detect if same code blocks exist in any other file.

VII. FUTURE WORK

The scope of this research was to implement the proposed methodology and validate the new approach with algorithms. The tool developed is a prototype with limited scope, but it can be further scaled up to a product.

A. The project may be scaled to detect clone code of other types of python syntax
B. Input criteria may also be scaled to multiple directories with python source files.
C. The functionality can also be extended with detection of TYPE-3 clones.
D. The project can also be scaled up to detect clone of other languages with the change of parser and AST generation component.

REFERENCES

[1] Baker, B.S.: A program for identifying duplicated code.de. In: Proceedings of ComputingScience and Statistics, 24th Symposium on the Interface, pp. 49–57, March 1993

[2] Baker, B.S.: Parameterized difference. In: Proceedings of the 10th ACM-SIAM Symposiumon Discrete Algorithms (SODA 1999), Maryland, USA, pp. 854–855, January 1999

[3] Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clonedetection system for large scale source code. IEEE Trans. Softw. Eng. 28(7), 54–67 (2002)

[4] Li, Z., Lu, S., Myagmar, S., Zhou, Y.: CP-miner: a tool for finding copy-paste and related bugs in operating system code. In: Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), Berkeley, CA, USA, vol. 4, no. 19, pp. 289–302, December 2004

[5] Li, Z., Lu, S., Myagmar, S., Zhou, Y.: CP-miner: finding copy-paste and related bugs in large-scale software code. IEEE Trans. Softw. Eng. 32(3), 176–192 (2006)

[6] Juergens, E., Deissenboeck, F., Hummel, B.: Clone detective - a workbench for clonedetection research. In: Proceedings of the 31st IEEE International Conference on Software Engineering, Vancouver, BC, pp. 603–606 (2009)

[7] Kawaguchi, S., Yamashina, T., Uwano, H., Fushida, K., Kamei, Y., Nagura, M., Iida, H.: SHINOBI: a tool for automatic code clone detection in the idea. In: Proceedings of 16th IEEE Working Conference on Reverse Engineering (WCRE 2009), Lille, pp. 313–314 (2009)

[8] B. Baker, A Program for Identifying Duplicated Code, in: Proceedings of Computing Science and Statistics: 24th Symposium on the Interface, 1992, Vol. 24:4957, 24:49-57.