iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** III **Month of publication:** March 2018

DOI: http://doi.org/10.22214/ijraset.2018.3748

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue III, March 2018- Available at www.ijraset.com

Java Source Code Plagiarism Detection System

Mrs Ghuge Madhuri Laxman[1], Prof V. R Chirchi[2]
[1]*PG Student, PG Department of MBES College of Engg. Ambajogai, BAMU University*
[2]*Assistant Professor, PG Department of MBES College of Engg. Ambajogai*

Abstract- Java source duplication is a main problem in academia. Many instituation systems mostly use plagiarism detection system or tools to find out similar source code files in a project or program. Same files are detected, the system process with the further investigation process which detect the similar source code files or assignment with the help of providing plagiarism. This system defines various tools that can be integrated with already defined plagiarism detection to improving plagiarism detection performance. The system can be improving new idea between source code files with big corpus files. The source code files are matched with file which are already define or save in big corpus data. This system can retriev data technique, with the help of detecting plagiarism using latent semantic analysis, is important within the specific files or big corpus.
Keywords - plagiarism detection, preprocessing, latent semantic analysis, cosine-similarity.

I. INTRODUCTION

In the computer science and technology, there are number of possibilities of code reuse problems are occurred. In most of educational system students submit their projects and assignment, there may be possibility of duplication in java source code. In these case source code plagiarism detection is difficult to check manually one by one. people uses programming assignment with copied form [7].The plagiarism detection uses two main process. In the first part, it generates a preprocessing from a given program. Preprocessing eliminates the blank spaces,comments,operators,symbols.The next step of representation is used for check the similarity between two programs or projects. A tokenization is most useful for intermediate representation.

Plagiarism detection system uses the tokenization sequence.Tokenization process different kind of keywords,identifiers,constants are seperated In the second part, system evaluates several techniques which are cosine similarity and term by file matrix methods are developed for identify the similar code in various files.Source code plagiarism is major problem and it takes several steps to detect. Usually, when students are solving the same problem by using the same programming language source code, there is a high possibility that their assignment solutions will be more similar. Result of such processing are regenerating & combining several segments copied from one or more source in assignment plagiarism [9].

In source-code files, similarity of codes are measured in terms of suspicious files or innocent files.similarities between source code files indicate it is suspicious file . Innocent due to source code is original in nature means all the files are original. plagiarized code are name as suspicious source code. Files are define source code fragments which means small modules that are different in program and their logic with contents and functionality from source-code fragments find in the main corpus database.

II. LITERATURE REVIEW

Brenda S. Baker.[2] This paper describe the duplication on large data set.these codes are free from plagiarism
To avoid the copied data.

T. Kamiya, S. Kusumoto, and K. Inoue[6]., Thid paper define tokenization with plagiarism detection for large source code. A code is a part of source files that is identical or similar to one or more code files. These modifications are difficult to find.

Jiang, Z. Su, and E. Chiu.[8] In this paper,different types of technique used to discover code related errors. It uses a efficient algorithm for detecting plagiarism with different stages. birthmark used to help them to define code theft by identifying different properties of a program.

L. Jiang, G. Misherghi, Z. Su, and S. Glondu.[9] . In this paper, efficient algorithm for identifying similarity between two source codes with proper representations of source code in various files.

III. ALGORITHMS

A. Latent Semantic Analysis

Latent Semantic Analysis is automatic statistical, mathematical technique and methods for inferring and extracting connection of contextual usage into programs. It reduces noise in data, it uses synonymy to calculate different term but same meaning or polysemy

to overcome problem with one term it takes more meaning .it is language independent not necessary to develop any parser.Parsed data convert words defined as unique character strings base separated into passages such as paragraphs or sentences.

1) Tokenisation.
2) Removal of Comment.
3) Remove all the numbers.
4) Removing non-frequent tokens.
5)Creation of union set of all token from each source code file.
6)File matrix generation.
7)Cosine similarity.
8)Collecting union of suspicius file and union
9)Report generation.

B. Source-Code Preprocess

Preprocessing is the first phase. In the first phase of preprocessing remove the source comments,blank spaces,special symbol. The source code spliting or merge of variable declarations , changing the order of variables as well as addition of some redundant statements are done.

C. Create term by file Matrix

The n-gram string-matching techniques, the corresponding similarity metric uses overlapping bi-gram string matching metrics such as the cosine similarity for calculating the similarity between the one or more files.

D. Similarity Measures

Similarity measures detect the similarity between two files which are the source code files already define and query define source code.these measure define its characteristics,and behavior of code. It is either suspicious files or innocent files. amount of suspicious similarity is found then the files under investigation can be considered suspicious. . All similarity between files detect very carefully to determine whether it is innocent or suspicious files.Using cosine formula measure the similarity between query vector and file vector.

E. Plagiarism Report

1) Set of suspicious and innocent file.
2) Pdf file creation.

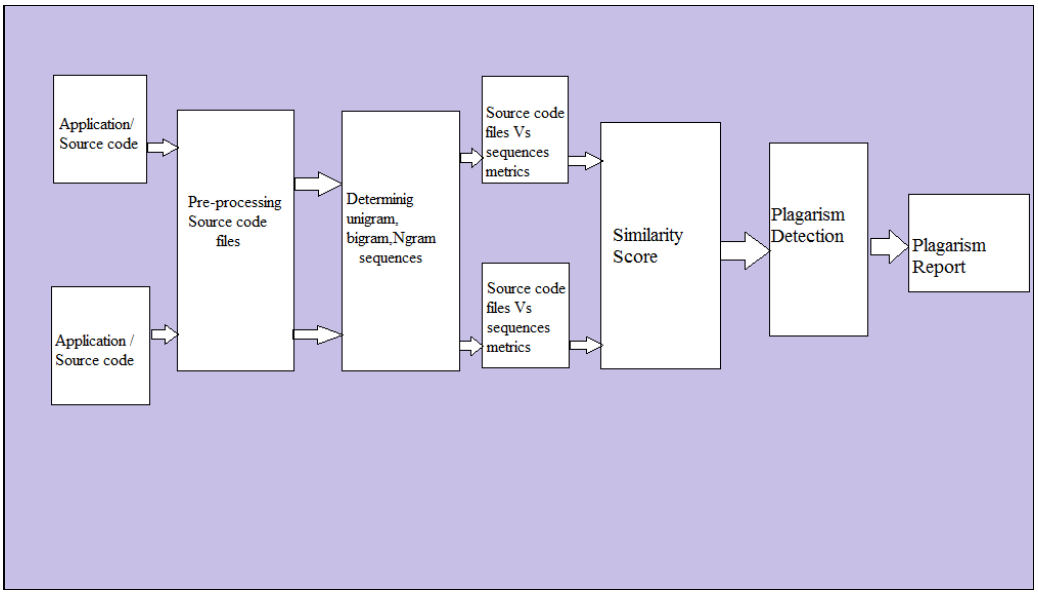


Fig 1 System Architecture.

IV. HARDWARE AND SOFTWARE SPECIFICATION

A. Hardware Requirements

1) Ram: 4GB or higher
2) Hard Disk: 40GB or higher
3) Processor: i3 or advanced

B. Software Requirements

1) Framework: NetBeans IDE, 6.8 or more.
2) Coding Language: JDK 1.6 or more.

V. ADVANTAGES

A. Improve plagiarism detection in educational system.
B. To increase the quality of programs and educational assignment. So these system is useful in acedima.

VI. OUTPUT DESIGN AND RESULT

Result of the system is tested over the privious code theft detection system. It is to store the data of privious student project as well as programming assignment. The java assignment and project source code uniquely identify. Newly assignment, source code are to comparing with the dataset.The system can be start to it enables to detection but stored data should be access or browse.The browsing data are comparing with source file. Browsing source file is comparing with dataset, but the preprocessing are done through this method. All comments, block comments, keywords, operators are removed. when the comparing file, all the tokens are collecting and stored.

Single file token collection.
Union set of all files token collection.

To detect Suspicious and Innocent files using cosine similarity method between query vector and source vector. Suspicious files contain copied data and Innocent files contains original code. Using suspicious files generate Plagiarism Report.Maximum plagiarism and total plagiarism.

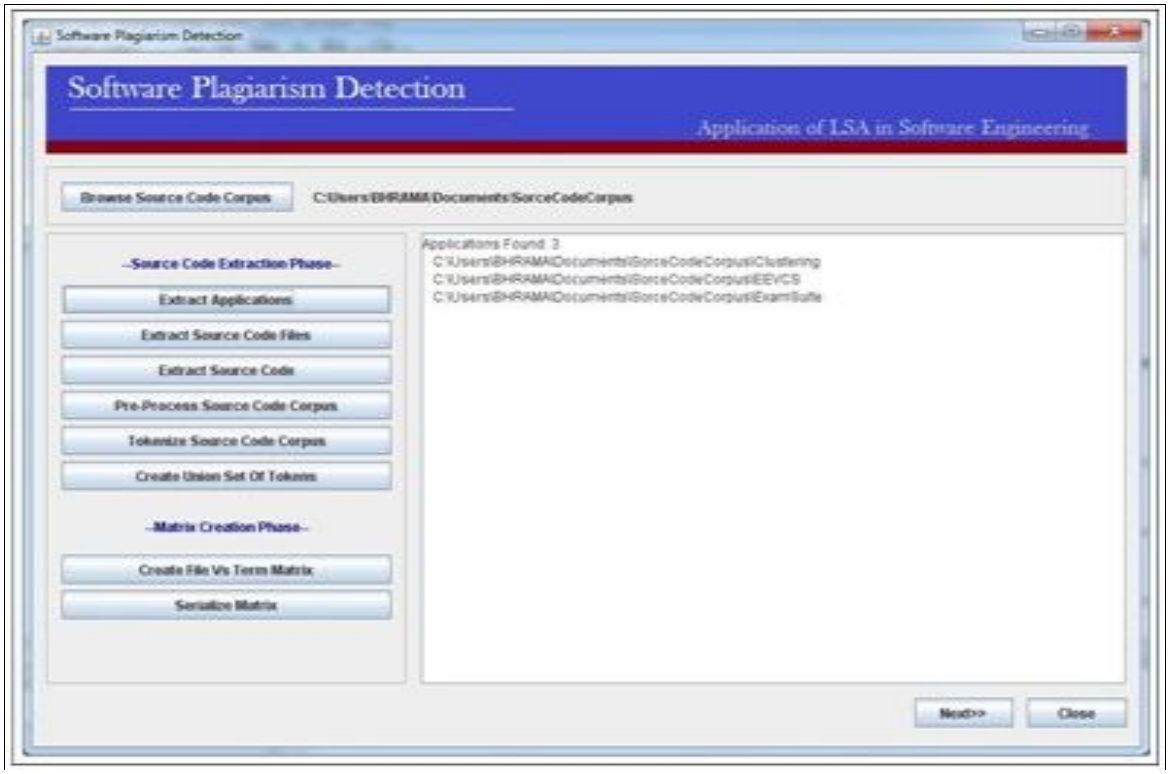


Fig 2. Plagiarism detection process.

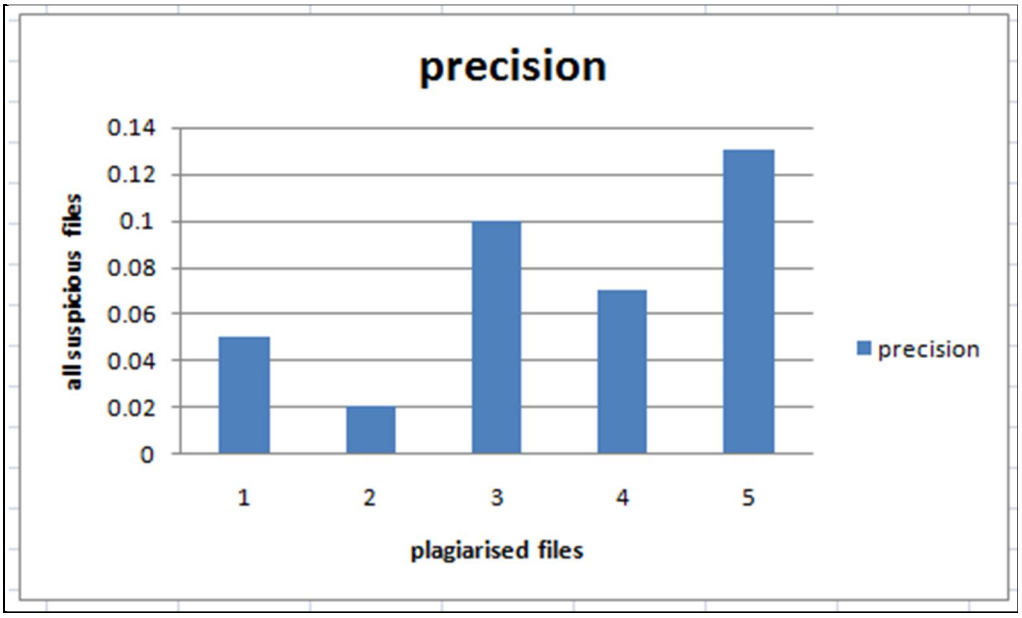


Fig 3. Graph of suspicious files.

Fig 3 determines the estimation of suspicious files out of all defined files. precision calculates how many files are suspicious or innocent in our data files. precision calculate with the following method.

Precision= |SD|
|DF|

number of suspicious file pairs detected

---

total number of file pairs detected

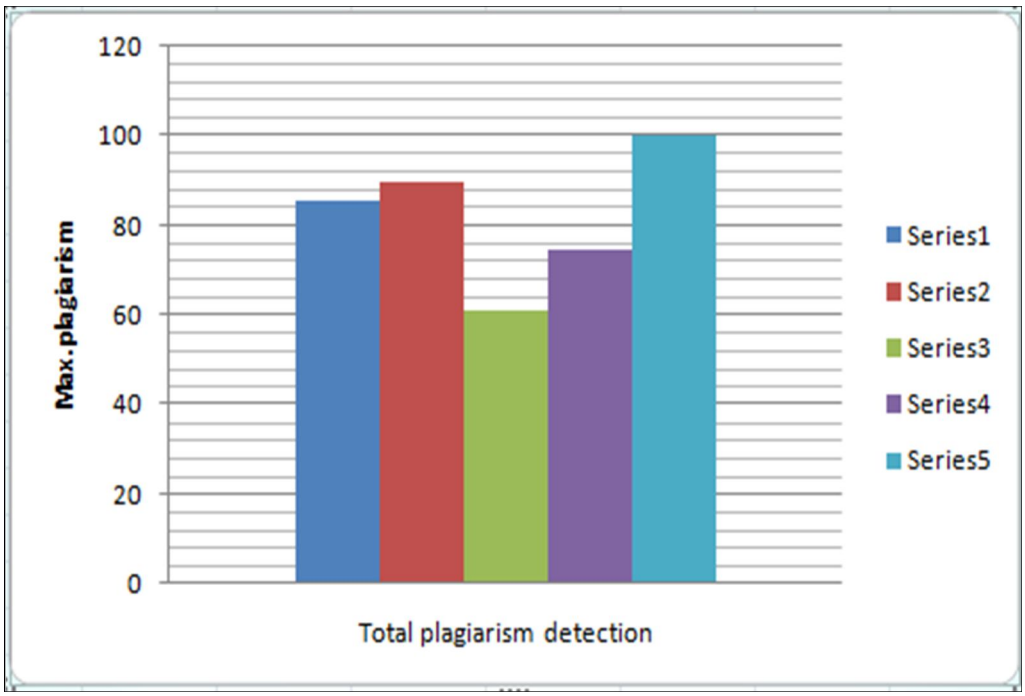


Fig 4. graph of total plagiarism in files.

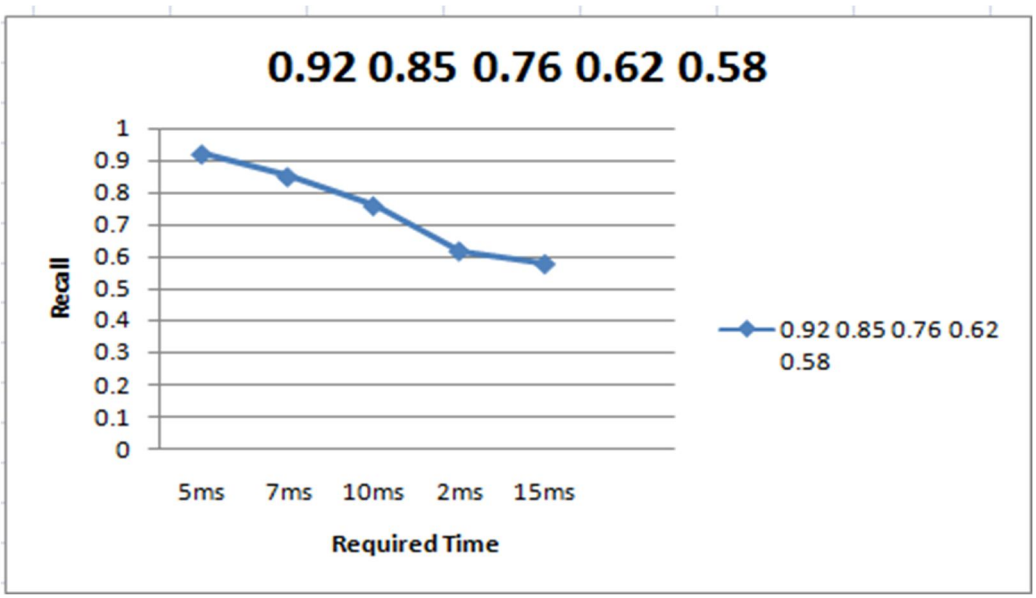


Fig 5.graph of Recall process in files.

Recall defines the calculation of number of suspicious file pair detected in source code in query vector out of suspicious file pair.These result determine the performance of plagiarism detection in source code file.

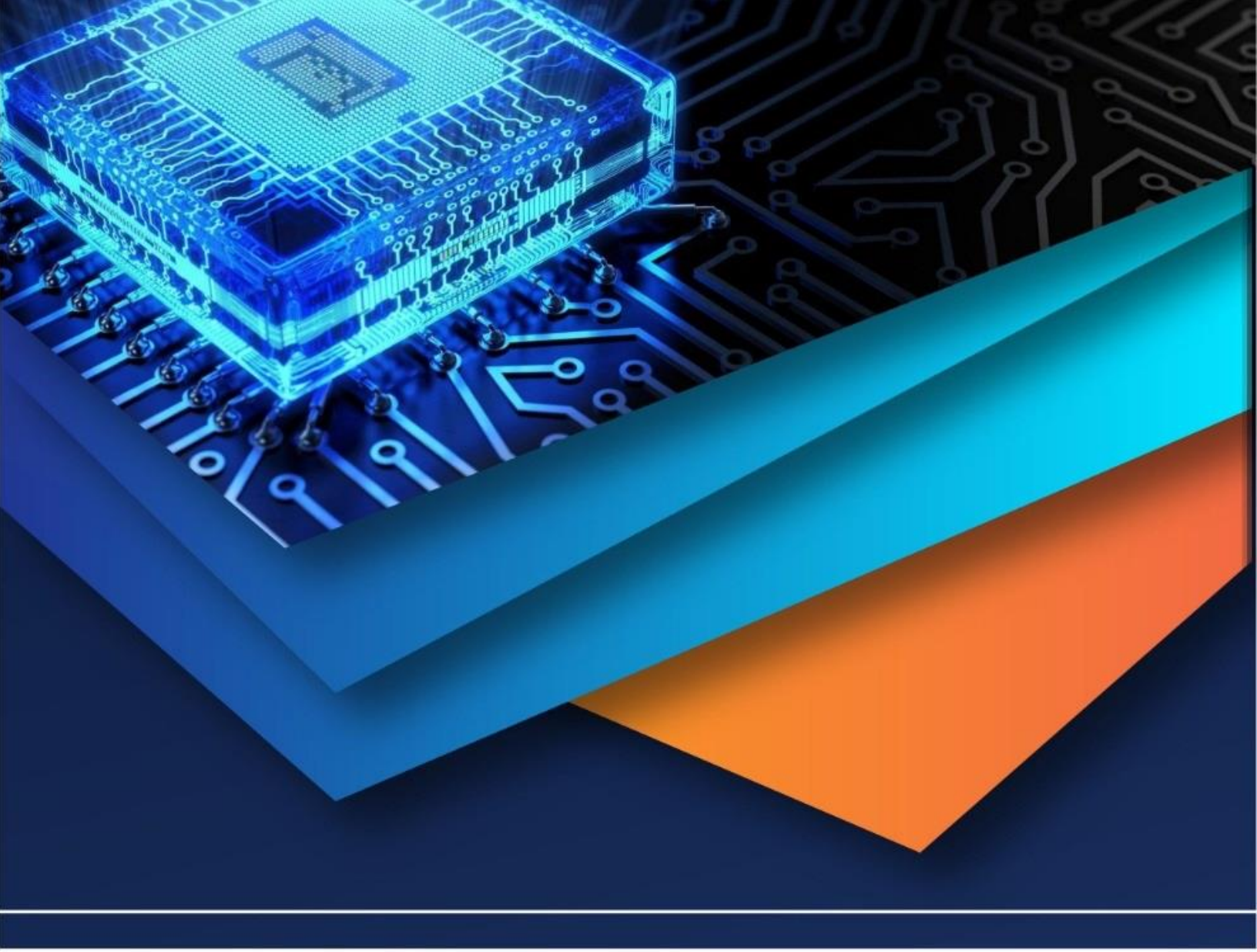
$$\text{Recall} = \frac{\text{number of suspicious file pairs detected}}{\text{total number of suspicious file pairs}}$$

VII. CONCLUSION

This result paper proposes a best approach based on the Latent Semantic Analysis information relevant method for the plagiarism detection and investigation process. This tool developed for java souce code to give better result for different source codes.it work separately or current plagiarism detection system.

REFERENCES

[1] Yoon-Chan Jhi, Xinran Wang, Xiaoqi Jia, Sencun Zhu,Peng Liu, Dinghao Wu, "Program Characterization Using Runtime Values and Its Application to Software Plagiarism Detection," IEEE TRAN, ON Software Engineering, VOL. 10.1109/TSE.2015.2418777.
[2] B. S. Baker, "On finding duplication and near-duplication in large software systems," in Proceedings of 2nd Working Conference on Reverse Engineering (WCRE '95), 1995, pp. 86–95.
[3] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees." in Int. Conf. on Software Maintenance, 1998.
[4] K. Kontogiannis, M. Galler, and R. DeMori, "Detecting code similarity using patterns." in Working Notes of 3rd Workshop on AI and Software Engineering, 1995.
[5] J. Krinke, "Identifying similar code with program dependence graphs." in Proceedings of Eighth Working Conference on Reverse Engineering (WCRE'01), 2001, pp. 301–309.
[6] T. Kamiya, S. Kusumoto, and K. Inoue., "CCFinder: a multilin-guistic token-based code clone detection system for large scale source code." IEEE Transactions on Software Engineering, vol. 28,no. 7, pp. 654–670, 2002.
[7] M. Gabel, L. Jiang, and Z. Su, "Scalable detection of semantic clones," in Proceedings of the 30th International Conference on Software Engineering (ICSE'08), 2008, pp. 321–330.
[8] L. Jiang, Z. Su, and E. Chiu, "Context-based detection of clone related bugs," in Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT symposium on the Foundations of Software Engineering, ser. ESECFSE '07, 2007, pp. 55–64.
[9] L. Jiang, G. Misherghi, Z. Su, and S. Glondu, "DECKARD: Scalable and accurate tree-based detection of code clones," in Proceedings of the 29th International Conference on Software Engineering (ICSE '07), 2007, pp. 96– 105.
[10] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," The Univeristy of Auckland, Tech. Rep.148, Jul. 1997.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY