International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 07, July-2016

Program Similarity Detection Tool

Prajakta Dhamanskar
Assistant Professor,
Dept of Information Tech
Fr Conceicao Rodrigues College of Engineering,
Mumbai, India

Kevin Garda
Student,
Department of Information Technology
Fr Conceicao Rodrigues College of Engineering,
Mumbai, India

Yash Sanzgiri
Student,
Department of Information Technology
Fr Conceicao Rodrigues College of Engineering,
Mumbai, India

Akshay Pujare
Student,
Department of Information Technology
Fr Conceicao Rodrigues College of Engineering,
Mumbai, India

***Abstract*—Plagiarism is copying another's work. Copying source code frequently happens in programming assignments in colleges. Direct copying does constitute academic dishonesty. So the student should be motivated to gain more understanding by typing independently than by copying code directly. The work in this paper presents a tool which would be used for detecting plagiarism in the programs performed by students. Also this would be used to detect whether the source code is copied from internet. Input to the tool would be the programs that need to be checked for plagiarism. The tool would give output as percentage of plagiarism detected between each pairs.**

Keywords— Plagiarism; Algorithm; Programs; Program Similarity; Karp Rabin; C Programming

I. INTRODUCTION

Plagiarism is an act of reusing someone else's work without the permission of original author.[1][6] A good example of plagiarism is source code plagiarism where a student submits a program whose part is copied from another student's program or the internet. Plagiarism in assignments is an increasingly common problem. Several surveys showed that a 46% of students have engaged in some form of academic dishonesty, particularly plagiarism.

Source code plagiarism is an easy to do task, usually, when students are solving the same problem by using the same programming language, the probability of their solutions looking same is high. Thus detecting Plagiarism is very tedious for the faculty members without proper tool support.

II. LITERATURE SURVEY

There are three main categories of plagiarism detection approaches:[6][7] text-based, attribute oriented code-based and structure-oriented code-based . These approaches are used by source code similarity detection tools that can be divided into two categories: offline and online. Online source code similarity detection tools can check a document for fragments

that can be found through web search engines, while offline source code similarity detection tools check for similarity between documents usually stored in a database.

Previous work has been done in the field of plagiarism as well as detecting similarity in programs. The methods used for detecting similarity are basically of 2 types

• Feature Comparison

• Structural Comparison

Feature comparison algorithms compare documents based on their attributes (for example, number of distinct tokens, overall word count, average number of characters per line) and compares the profiles of submissions to determine if they are unusually similar. Structural comparisons compare the content of submissions for example, comparing the specific tokens that form the inputs.

Some Plagiarism detection tools and algorithms used are discussed below.

A. Moss (for a Measure of Software Similarity)

MOSS[3] [4] [5] is an automatic system for determining the similarity of programs. The main application of Moss has been in detecting plagiarism in programming classes. MOSS was developed in 1994, and has been very essential in detecting similarity in programs. The algorithm behind moss is a significant improvement over other cheating detection algorithms

MOSS is not a system to completely detect Plagiarism. It is still up to a human to go and look at the parts of the code that Moss highlights and make a decision about whether there is plagiarism or not. One way of thinking about what Moss provides is that it saves teachers a lot of time by pointing out the parts of programs that are worth a more detailed examination. Someone must still look at the code. Moss is being provided as an Internet service. The service has been designed to be very easy to use–you supply a list of files to compare and Moss does the rest. The current Moss submission script is for Linux.

B. JPlag

JPlag is a web service developed by Guido Malpohl in 1996. JPlag compares closely with the performance of MOSS. In JPlag users submit an archive of programming[11]

files to the web service of JPlag; the files are then compared pairwise against each other. Jplag first runs the programs through a parser or scanner for the appropriate programming language, then uses the outputs from the parser or scanner as the input strings to its backend algorithm; only the front-end parsing step is language-dependent . JPlag supports C, C++, Java, Scheme, and even natural language.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 07, July-2016

C. Karp Rabin Algorithm

Karp-Rabin Algorithm [1] is a string matching algorithm. It uses fingerprints to find occurrences of one string into another string. Karp-Rabin Algorithm reduces time of comparison of two sequences by assigning hash value to each string and word. Without hash value, it takes too much time for comparison like if there is a word W and input string is S then word is compared with every string and sub string in program and hence it consumes more time. Karp-Rabin [1] has introduced concept of Hash value to avoid time complexity O(m2). It assigns hash value by calculating to both word and string/substring. So hash of substring (S) matches with hash value of W then only we can say exact comparison is done.

III. BLOCK DIAGRAM

The Project flow is as following [7]

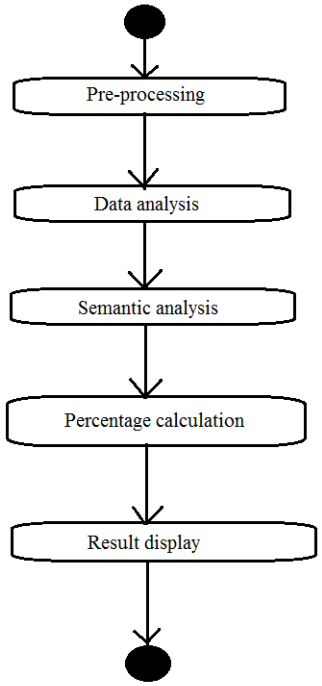

Fig. 1. Block Diagram

A. Preprocessing

This is the first phase, during this step, all comments and white spaces are removed from the original source code.

B. Data Analysis

This stage involves checking, the number of data types, number of identifiers and number of operators i.e. +,-,*, / are checked. If the number is same in both the files, then the programs are 100 % copied.

C. Semantic Analysis

During this phase the various semantics are analysed. The expressions are checked if they are semantically equivalent or not. The Logic discussed in section 6.1.2 is used to decide
if the 2 expressions are similar

D. Similarity Measurement

This is the part where the copy percentage is assigned to the 2 programs using the algorithm as discussed below.
Percentage Match calculation Algorithm:-
If the contents of the files are same or if the content of the file follows the following test cases:-
i) Changing Identifier
ii) Replacing expressions with semantically identical equivalents
Then percentage=100%
Else
{
If number of data types is same in both the files
then
percentage =+15
If expressions in both the files is same
Then
percentage=+15
If number of '+' operators are same in both files
Then
percentage=+14

If number of '-' operators are same in both files
Then
percentage=+14
If number of '*' operators are same in both files
Then
percentage=+14
If number of '/' operators are same in both files
Then
percentage=+14
}

E. Final Similarity Calculation

In this stage the final results are displayed to the users

IV. IMPLEMENTATION DETAILS

The following section explains the various methods and algorithms used for solving the test cases.

A. Removing Coments and Whitespaces

In order to remove white spaces and comments, regular expressions are used
The notations of the various regular expressions are:
[\t] => for whitespace or tab
* => for start of multiline comments
*? => for anything followed by new line
\r => carriage return in ASCII (in simple word :- to avoid junk character)
\n => for linefeed
1. Removes multi-line comments and does not create a blank line, also treats white spaces/tabs
$text = preg_replace('!^[\t]*/*.*?*/[\t]*[\r\n]!s', '', $text);
Note:-> if multiline comments start with or without tab or whitespaces is taken care by ^[\t]* then it will check for multiline comments to start by matching * then it will check for anything either followed by new line or not by matching *? Then it will find the end of comments by matching */ then it will check is there any whitespace or tab present or not by matching [\t]* till new line by matching [\r\n]!s then replace with nothing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 07, July-2016

2. Removes single line '//' comments, treats blank characters $text = preg_replace('![\t]*//.*[\t]*[\r\n]!', '', $text); Note:-> if single line comments start with or without tab or whitespaces is taken care by ^[\t]* then it will check for single line comments to start by matching // then it will check is there any whitespace or tab present or not by matching [\t]* till new line by matching [\r\n]!s then replace with nothing

3. Strip blank lines $text = preg_replace("/(^[\r\n]*|[\r\n]+)[\s\t]*[\r\n]+/", "\n", $text)

B. Replacing with semantically equivalent

In order to check if the 2 expressions are same, we replace one of them with their semantically equivalent statement and then compare the sentences. The following are the replacements:-

- <= with !(>) or vice-versa
- < with !(>=) or vice-versa
- > with !(<=) or vice-versa
- >= with !(<) or vice-versa
- 1 with !0 or vice-versa
- 0 with !1 or vice-versa

These can be used in the conditions in the if else statements, for , while loops and do while.

Basic Algorithm which is applied in expression containing '!'(not operator):-

1)Find position of '!'
2)Remove character before '!'.
3)Remove '!'.
4)Replace :-
i)!((>)) with <=
ii) !(>=)with <
iii) !(<=) with >
iv) !(<) with >=
v)1 with !0
vi)0 with !1
5)Remove first occurrence of ')'.

C. Changing Identifier

Initially in order to detect the change in identifiers, tokenization was proposed. Tokenization [7] is the process of converting the source code into tokens. This technique is very popular and used by many source code plagiarism detection systems. But because of its growing complexity, a new method has been proposed. The algorithm of the new method is as given below:-

Algorithm
Input:-File1, File 2
Step1: Count number of data types and operators in file1 and file2.
Step2: If the number of data types and operators are same, that means the identifiers have been changed but logic is same.

V. INTEGRATING INTERNET CHECKING

As discussed earlier, Plagiarism can be done using various techniques. Since programs can be easily found on the internet, the software would check for program similarity by comparing the given programs with the internet programs. For this it would make use of the Google API.

The steps to implement that are as follows:-

1. Visit https://cse.google.co.in/cse/create/new
2. Click on create.
3. Click on Control Panel.
4. Click on Add
5. Click on search engine id
6. Click on Update
7. Go to https://console.developers.google.com/start
8. Click on Enable and Manage API

9 Click on the project you have created.

8. Go to custom search API
9. Click on enable
10. Click on credentials
11. Click on add API key. Then on Server key.
12. Finish

VI. FINAL RESULTS

The tool was tested against certain test programs. The topics of these test programs were taken from the list of practical programs assigned to the students. The results of our tool are compared to the results of by the feature based approach.

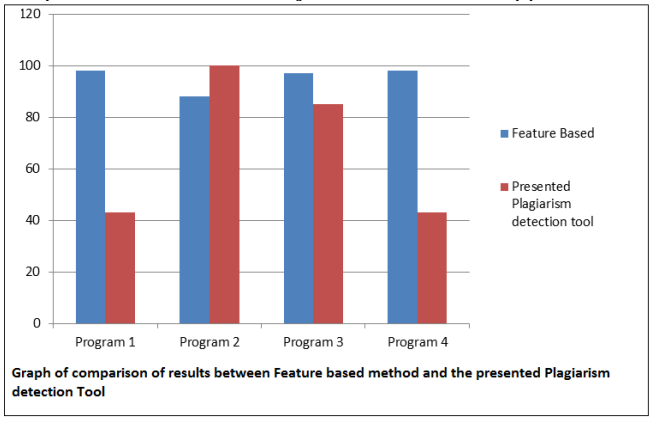

Fig. 2. Comparison of the results

The graph above shows the comparison of the results shown by this tool and the feature comparison method (Attribute counting method) to detect plagiarism in code. In the attribute counting systems programs are depicted by various quantities such number of distinct tokens, overall word count, and average number of characters per line. Then the similarity between two programs is calculated by comparing their respective values. By comparing the results of these methods with the results of this tool some disadvantages of the feature based method can be seen.

In Program 1, the logic used by the 2 programs to solve the problem is completely different. But since their features (word count, characters) are nearly same, the feature based method calculates 98% similarity. This tool on the other hand analyses the code and denotes 43% similarity.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 07, July-2016

In Program 2, Some minor changes are made to the 2 programs i.e. Addition of extra variables, changing names of identifiers .Hence the codes are completely same and this tool assigns 98 % similarity (close enough to 100% plagiarised). Feature based approach gives 88% similarity which is although high but not accurate enough.

In Program 3, although the conditional loops are modified the features of both the programs are same, thus it is assigned 97% similarity as per feature based approach. This tool assigns a similarity percentage in accordance with the modification i.e. 85%

In Program 4, The 2 Programs have completely different logic but share some similarity in the declarations and Language syntax. This tool assigns 43% similarity which is comparatively less to be considered for Plagiarism. Feature based approach gives 100% similarity.

Thus feature based approach has a disadvantage that it can be either very insensitive (two programs might share the same measures while they completely differ in the logic) or very sensitive as it ignores the programs structure [10]

CONCLUSION

The tool presented here, provides instructors with a simple interface that allows programming instructors to rapidly check assignments for academic dishonesty. The instructor simply uploads the programs and can get the percentage match between them as output. The current version is limited to C programming language. It has been successfully tested on old assignments, and shows accuracy when compared to the feature based method of Plagiarism detection.

REFERENCES

[1] P. S. Sonawane Kiran Shivaji, "Plagiarism detection by using karp-rabin and string matching algorithm together," *International Journal of Computer Applications*, vol. 11 6- no 23, pp. 0975– 8887, April 2015.

[2] G. Whale., "Identification of program similarity in large populations." ." In: The Computer Journal 33.2 (1990), pp. 140–146.

[3] A. Aiken. A system for detecting software plagiarism. [Online]. Available: https: //theory.stanford.edu/~aiken/moss/

[4] A. A. Saul Schleimer, Daniel Wilkerson, "Winnowing: Local algorithms for document fin-gerprinting."

[5] K. W. Bowyer and L. O. Hall., "Experience using moss to detect cheating on programming assignments in: Frontiers in education conference, 1999. 29th annual. vol. 3. ieee. 1999, 13b3âĂß18." *Frontiers in Education Conference*, vol. 3, 1999.

[6] Vreda, "[automated assessment of programming assignments]," 2013.

[7] D. G. Zoran Djuric, "A source code similarity system for plagiarism detection."

[8] Yash Sanzgiri,Kevin Garda,Akshay Pujare,"Plagiarism Detection tool for Programs"IJARCCE , Nov 2015

[9] K. L. Verco and M. J. Wise, "Software for detecting suspected plagiarism: Comparing structure and attribute-counting systems." *Proceedings of the First Australian Conference on Computer Science Education*, pp. 81–88, 1996.

[10] M. Heon and D. Murvihill, "Program similarity detection with checksims."

[11] L. Prechelt, G. Malpohl, and M. Phlippsen, "Jplag: Finding plagiarisms among a set of programs."