C CODE PLAGIARISM DETECTION SYSTEM

N.HARITHA (MTECH CSE)\hspace{0.5cm}M.BHAVANI ASST.PROFESSOR\hspace{0.5cm}K.THAMMI REDDY PROF

VIIT, VISAKHAPATNAM, AP, INDIA\hspace{0.5cm}GITAM, VISAKHAPATNAM, AP, INDIA

ABSTRACT - This paper aims at designing a new code plagiarism detection system for programs written in C language. Plagiarism in academics computing courses has become a widespread problem. Various tools like jplag moss etc fail to identify some of the modifications made by the students to disguise plagiarism like adding redundant statements etc. So taking into account the need for a good plagiarism system the following system is designed. The following system can used as an assistance to the instructors to measure the performance of the students.

The main advantage of this system is it gives the user with two options like checking file or checking folder and gives a pictorial representation of the result.

Keywords: Plagiarism, interlingual plagiarism, intralingual plagiarism, attribute comparison, structure comparison, tokenization, n-gram technique, jaccard’s coefficient.

INTRODUCTION

Plagiarism is defined simply as imitation or simply copying of somebody else’s work and pretending as if it is their own work. Using one’s work with out proper citation becomes plagiarism. We have to include the source from which the text or idea has been used.

Plagiarism occurs in two forms, it can be either text plagiarism or code plagiarism. There has been a substantial research done on text plagiarism so now the focus is mainly on code plagiarism. Code plagiarism is nothing but copying a program’s logic and strictly speaking a program’s structure from some one with or with out his permission.

Now a days, technology has brought world to our feet, search engines have made our life pretty easy, where we can found solution for anything and everything. Technology has not only made our lives simpler but also made it miserable. People are becoming lazy and students, they prefer to spend more time in social networking than to complete their assignments.

Most of the institutions are assessing students performance through assignments only.

Students try to copy from their fellow students when they lack the ability. But there are also some cases where students under pressure knowingly and unknowingly try to copy from their friends, the pressure comes from their instructors giving them a deadline for submission of their assignments or threatening them by punishment or marks.

So, the main difficult task is now faced by the instructors themselves because they have to give marks to the students basing on the performance of the students in the labs. It becomes a tiresome and time taking job to compare each and every student’s work with the others to find out who has copied. And when the number of students in the class becomes more, this work becomes a very difficult task.

So, this leads us to go for a good software which can compare all the assignments of all the students and give us a good representation of the result.
So the requirement is a good automated code plagiarism detector.

There are many automated plagiarism detection tools available like JPlag, moss, yap, plaggie, fpds etc. Some of them are free tools and some need an account to be created.

Basically what a code plagiarism detector does is, it is comparing the programs for similarity.

Code plagiarism occurs in two forms, they are

1. **Inter Lingual Plagiarism**: code written in one language is plagiarized.

2. **Intra Lingual Plagiarism**: code written in one language is plagiarized in another language.

We are now here talking about inter lingual plagiarism. We have two main methods of comparison.

### 1. Attribute comparison

As the name suggests, attribute comparison is nothing but the comparison of the number of attributes like number of variables and number of constants and performing a comparison. The following parameters called the Halsted’s parameters are used to perform the comparison.

- \( N_1 \) = number of operators
- \( N_2 \) = number of operands
- \( n_1 \) = total number of operator occurrences
- \( n_2 \) = total number of operand occurrences

The metrics formed using these parameters are

\[
V = (N_1+N_2) \log_2 (n_1+n_2)
\]

\[
E = [n_1N_2(N_1+N_2)\log_2(n_1+n_2)] = (2n_2)
\]

Submitted programs having the same value of \( V \) and \( E \) are found to be suspicious.

This method of comparison is simple to implement; however, we are not giving importance to the structure of the program.

### 2. Structure comparison

Structure comparison takes into account the structure of the program and performs the comparison. We represent the structure of the program in some intermediate form and perform the comparison. The whole process takes place in three different phases.

- The first phase is the representation of structure in some intermediate form. There are many representations like tokens, parse trees, graphs, matrices, etc. Each has their own advantages and disadvantages.
- In the second phase, we apply some algorithm and find out the similarity ratio. They may either be string matching based algorithms or parameterized algorithms or parse tree comparison algorithms.
- In the final phase, the results of the comparison are represented in some visual or textual representation.

We have to take into account the various modifications that can be made on the programs in order to disguise plagiarism.

The modifications can be either textual or functional.

Textual modifications are done by simply changing the names of variables or...
changing loops, adding unnecessary useless statements etc.

Functional modifications are done only when the modifier has a clear idea of the logic, like generating new functions from the code etc.

**DESIGN**

In real an instructor after following many methods like changing the lab assignments every time a new course is designed, or by changing the programs every time a new semester is stated won’t help to prevent plagiarism.

This system is mainly designed for C programs. We can use this system to compare a given program with a set of programs i.e.) local database to find out the similarity. It can also be used to check a given a folder of files, to find out the suspicious and non-suspicious programs.

Output representation plays a very important role in this system. Visual representation plays a very good role compared to the textual representation. This system outputs the results in the form of graphs and tables.

**IMPLEMENTATION**

The main phases in the implementation of this detection system are as follows.

1. **First phase:** Tokenization is chosen as the intermediate form of representation of the structure of the program. Tokenization helps to detect the plagiarism disguises like changing of names of variables and changing loops.

2. **Second phase:** In the second phase, the tokens formed are represented by an alpha numeric character to form a simple sequence there by preserving the structure of the program. Finger prints as created using n-gram technique.

N-gram technique fights efficiently against adding redundant statements in code, n-grams are calculated like shown in this example

Suppose the sequence generated is

MDPSPSCPCPCP

The 3-grams generated (by sliding window containing 3 tokens at a time) will be

MDP DPS PSP PSC SCP CPC PCP CPC PCP

So additional statement will only affect the neighboring ngrams.

3. **Third phase:** similarity is calculated by using JACCARD’S SIMILARITY coefficient

Suppose program p1 has the following set of n-grams.

p1={MDP}, {DPS}, {PSP}

And program p2 has the set of n-grams as

p2={MDP}, {DDP}, {DPS}

Jaccard’s similarity coefficient (p1, p2) =

\[
\frac{|p1 \cap p2|}{|p1 \cup p2|}
\]

=4/8=50% similarity

It is necessary to have a threshold to know that the programs are plagiarized or not. In general 90% will be taken as the threshold and the programs having the similarity
below this threshold will be considered as non-suspicious and the others suspicious.

OUTPUT REPRESENTATION

This system offers two facilities to the user, whether one can use this to compare a given program with the programs folder in the local database to find out the similarity percentage. In the other way we can use this system as a whole to check the students performance on the whole by making a quick comparison.

The output representation here we are providing is the graph, where x-axis represents the programs in the folder used for comparison and y-axis represents the percentage ratio.

This provides a clear idea of how much similar the given program is with all the other programs in the folder.

![Graph](image)

By just analyzing the graph one can easily find out the similarity percentage.

In the second way we give the system a folder containing the assignments of the whole class to the system. The output will be given by calculating the similarity between each program with all the programs in the folder.

For the quick and easy analysis of the results we have taken the pictorial representation of tables.

The algorithm is taken as follows.

Step1: Start with program p in the folder, compare it with all the other files in the folder and find out the similarity percentage.

Step2: The similarity percentages are compared with the threshold.

Step3: Programs having the similarity percentage greater than threshold come under first group of suspicious programs and the other go to the non-suspicious group related to p. Remove p from the folder.

Step4: If the folder empty, yes the quit. Otherwise repeat the steps 1, 2, 3 with the next program in the folder.

Finally we get 2 blocks. First one showing the groups of suspicious programs.

The second one showing the non-suspicious programs.
ANALYSIS

This system is working efficiently given a large database and the results are coming clearly and fast. The main advantage of this detection system is one can easily analyze the results with the visual representation.

And one more thing is, if one program is added to the folder after some time, without having to repeat the whole procedure of comparison, we are giving the user with two options. To check file with a given database or to check a folder of files within themselves. The final analysis will be always done by the instructor by clearly observing the results.

Comparing with jplag, jplag gives the results in a tabular form. There is no pictorial representation and it is required that every file should be parsed before comparison. This system overcomes the disadvantage in jplag and provides a better visual representation that helps the instructors with fast and easy analysis.

REFERENCES


[12] Steven Burrows 1, Seyed M. M. Tahaghoghi 1 & Justin Zobel 1 “EFFICIENT AND EFFECTIVE PLAGIARISM DETECTION FOR LARGE CODE REPOSITORIES”


