SOURCE CODE PLAGIARISM DETECTION ENGINE
(CODEVISION)

Saeed A. Anabtawi , Basil Hassan

Supervisor: Dr.Othman Othman

CS department, An-Najah University, Palestine

ABSTRACT

In this project, we used different source code plagiarism detection methods such as Structure-based and keyword based to compare the accuracy and their ability to solve the problem. We build a modular scalable system that clusters and detects plagiarism in a visible form within a corpus of source files and test it aginst existing programs such as JPlag.

Keywords: *Similarity Detection, Jaccard, Longest Common Sub-sequence, Weighted Directed Graph, Clustering, Code Plagiarism.*

1. INTRODUCTION

Source code plagiarism is the act of copying code from others without giving any credit to the original programmer.The problem of code plagiarism could be in any education institute that teaches how to code and this problem affect the quality of educational in this institute. Moreover graduates of this institute will lack honesty and skill. This will lead to a major education problem, that is very necessary to be solved. Furthermore, the internet access became much easier in addition to the large number of programming code available on the Internet. All of this lead to creating different techniques to detect plagiarism in the source code. But the problem is the evaluation results might become misleading and unreal due to the plagiarism problem. Manual detection was found to be inefficient but it is effort and time consuming due to the vast amount of contents available, an automated plagiarism detection system becomes essential. “The definition of Plagiarism in software: a program which has been produced from another program with a small number of routine transformation challenge is to detect the techniques that the implicated students tend to use to disguise the copied code in order to mislead the grader”(A.Jadalla and A.Elnagar, 2007,[1]). Types of plagiarism: Intentional plagiarism is using other people work without any acknowledgment. Unintentional plagiarism the work is incidental but the authors are different and they used the same logic. Most common disguises:

- Changing comments.
- Changing identifiers.
- Changing the order of operands in expressions.
- Changing data types.
- Replacing expressions by equivalents.
- Adding redundant statements.
- Changing the order of time-independent statements.
- Changing the structure of iteration statements.
- Changing the structure of selection statements.
- Reordering of the code.

2. LITERATURE REVIEW

Plagiarism detection systems are classified into two main categories :

- The attribute-counting based (feature-based) systems :counting number of operands, operators, control statements, loop statements, conditional statements, and variables, to compute similarity, feature-based systems are more efficient and have relatively lower performance, but the problems are: It ignores the program’s structure. Two programs might share the same measures while they have completely different logic. It can hardly have very good performance because it throws away too much structural information.

- The structure-based system: have better performance and are less efficient, it compares the structures of two programs directly, It shows high performance in detecting source-code plagiarism, it uses four common techniques: String matching, Abstract syntax tree, Program dependence graph, Tokenization.

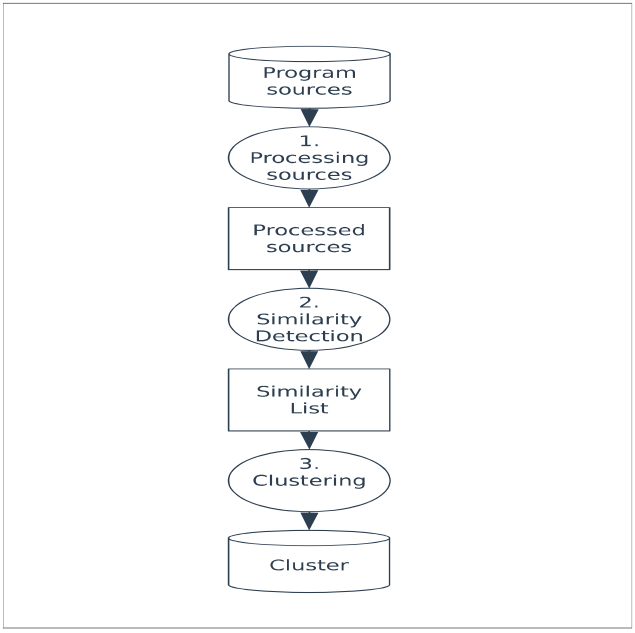
The first known plagiarism detection tool was an attribute counting program developed by Ottenstein. It uses the basic Halstead metrics to compare two FORTRAN programs. There are several examples of source code plagiarism tools. Focus in this section will be on: JPlag, SIM, and MOSS as a sample.

• JPlag: “Its a web-based plagiarism detection tool, its available for free and easy to use”[2].

• SIM: This is a tool that is developed to detect c code plagiarism, text plagiarism and DNA string comparison. The output is a similarity score value between 0 and 1 based on the level of similarity between codes.

• MOSS: its a popular free code plagiarism tool. It supports different operating systems. The tool divided the code into several finger prints and matching or similarity is evaluated based on the number of similar finger prints between the evaluated codes.

3. SYSTEM PIPELINE



4.1. Processing Sources

The first phase in the system is to load program source files, then process and apply multiple processing operations like filtration which includes filtration of noise elements such as comment, includes,...etc that could affect parsing operation that delivers a specific format for the next phase.

The filtration is done by regular expression. It handles and solves most cases of noise elements such as a comment written in a string, comment written between variables, includes any element that does not affect code logic.

After filtering program source files into a clean format the system performs a parsing and tokenization operations, this parsing and tokenization will generate a specific format that depends on the second phase, formats like AST (abstract parsing tree), a run list of tokens, list of keywords, a mapped abstracted code. The second phase similarity detection is a phase when the system detects the similarity between program source files using similarity detection method.
The input is provided as a set of source code files. Program files can't be compared directly because of noise elements such as useless “headers and include”, comments and blank lines, comments could contain fake codes which effect plagiarism detection system. Each comparing operation requires a specific format.

4.2. Similarity Detection

Each method takes a different format, The keyword-based method takes a list of keywords, The code is tokenized and transformed into a list of keyword. Then after that, a Jaccard similarity measure is applied on two lists of keywords each list represents a program and the result is a ratio between 0 and 1.

The Structure based method takes a mapped code as input. Then apply a modified Jaccard rule to compare 2 strings those strings are the mapped code of each program. The output of phase (2) is a similarity list that represents similarity between each file in the corpus. In phase (3) The similarity list is transformed into a weighted directed graph to apply the weighted graph clustering algorithm and perform weighted cuts. The system output is a cluster and it represents plagiarism in a form of groups

4.2.1 Keyword based

The output of the parse tree gets transformed into lists of keywords, which represents the program logic and structure. And similarity is computed using this statistical formula. The result is a ratio 0 – 1. If p1, p2 are two programs and T (p) is the indexed set of substitute keywords of program p [3], like T(P1) = [int1, int2, float1, for1, if1] a similarity measure we called the Jaccard coefficient [3] is defined by:

$$w(p_1, p_2) = \frac{|T(p_1) \cap T(p_2)|}{|T(p_1) \cup T(p_2)|},$$

4.2.2 Structrue based

It transforms the code to a sequence of symbols and character, That define program structure and components such as (type, class, objects,…..etc) and it gets mapped into a sequence of characters and symbol without any spaces [4]. We transfer each program into the signature from the run the formula, similarity between s1 which is program 1 and program will equal the length of longest common sequence times 2 divided by the length of s1 + length of s2. The result is a ratio between 0 – 1.

$$sim = \frac{2 * \text{lcs\_len}}{|s1| + |s2|}$$

The first step is to transform the source code into a sequence of well-defined identifier tokens[4]. The identified tokens are STRUCT: user defined struct type, different from the keyword, FUNC: functions defined in the program, ID: data variables, CON: constants, ARR: array, the keywords is still written as they are.

After that the system will insert them into mapping table, the output is a signature that represents the structure of the program.Then run the equation below on every 2 program signature:

```
diI=C;iF(iI,iA[],iI);vF(iI){iA
=C;g(iI=C;I<I;I++){iI=(A[I]*I)
F(C,&I);A[I]=I;}g(I=C;I<I;I++)
```

We have implemented the algorithm using the numbers instead of character to improve scalability and modified the LCS to work on an array of numbers instead of characters. to improve the scalability of the algorithm .

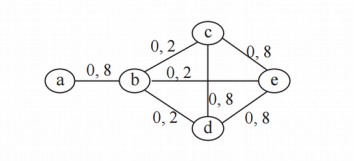
4.3. Clustering

3.3.1. Weighted Directed Graph

we construct a weighted non-directed graph G = (V, E) such that vertices V represent program identifiers and weighted edges E represent the similarity (evaluated by the Jaccard coefficient) between programs. In this graph we take into account program pairs only if their program members are associated with a Jaccard coefficient value higher than a considered cutoff criterion value (threshold) [3] , sim(p, p) = 1 and sim(p1, p2) sim(p2, p1) from those two rules the number of edges in the graph equal [n[n-1]]/2 [3].

Graph representation of similarity list

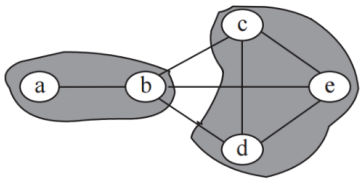
a	b	0.8
b	c	0.2
b	e	0.2
b	d	0.2
c	e	0.8
d	e	0.8
c	d	0.8



4.3.2. Grouping

The grouping algorithm (weighted graph clustering algorithm)[3] takes the result of the previous operation that outputs a weighted graph. Then using threshold and the clustering algorithm it generates groups or clusters from the weighted graph like in the next figure

Note: threshold = 0.80, the grouped nodes are shaded in gray



In the Figure, the weighted directed graph after clustering, in the beginning, it was clustered into one cluster, whereas the graph clustering algorithm optimizes the weighted edge connectivity and clusters and split it into two clusters. Although node (b) is connected to three nodes (c, d, e) but the weight of each connection is under 0.8 and the connection between the node (c, d, e) is over 0.8,therefore (c, d, e) is a cluster, whereas the weight of edge (a, b) is 0.8,(a, b) is another cluster. Thus, we may

consider that edges (b, c), (b, d) and (b, e) have been eliminated by clustering. Since these edges represent similarity between programs, their elimination facilitates in some cases the discovery of false detection's. The algorithm initially assigns a cluster to each node of a graph and proceeds to merge nodes to clusters according to the weights of their edges. The algorithm terminates when no further merging is possible.

The clustering algorithm (weighted graph clustering) is able to divide the results into groups, This feature helps to identify plagiarism in a more visible way. The threshold is the main component of the clustering algorithm if the system set it to low the clusters would become noisy, having too many elements and it will affect system accuracy. If it's too high, some source codes can't be detected. The system must analyze the data first and find the maximum similarity, minminm similarity and the average similarity (MMA model), this analysis well aid the process of selecting the right threshold for a given dataset.

4.3.3. Min-Max-Avg model (MMA)

The model we propose helps specify the value of the threshold by analyzing each data set we have. because every data set is unique there is no threshold work perfectly for each data set.After we process the data into different stages in system pipeline the result from stage number 2 is the similarity list. we analysis the similarity list to extract (min similarity, max similarity, average similarity).

The value of the threshold belongs in the domain [avg, max], theshhold>min, threshold < max. set the value below Avg will result of noise cluster that contains too many nodes, Avg: average similarity represent the normal percentage required to write a solution to solve the same problem if the question was to simple to solve the average would be very high for example 90% of data we tested on we set threshold to 99% to be able to cluster the dataset we have . otherwise, all nodes would be in one cluster. The MMA model helps to specify the domain to select the threshold from. it save up the time required to search entire domain to smaller domain depending on the data set given.

5. EXPERRMENT

We implemented the last two algorithms in python and using pycparser[5] to handle c99 programming language and make some experiments in order to choose the best of them.

5.1. Trial Experiment (Artificial dataset)

In this experiment we have a set of three problems, each one has a set of solutions (code sources).We generate some solutions from original ones as they been plagiarized. Then we test similarity detection algorithms. The perpouse of expermint is test main futuers of the system such ass accuracry and speed.

Experiment 1: 16 solutions included the 7 plagiarized ones & similarity threshold 80%.

	Structure based	Keyword based	
# of correct guesses	6	3	Higher is better
# of incorrect guesses	0	1	Lower is better
# of gross incorrect guesses	0	1	Lower is better

Experiment 2 : 16 solutions included the 7 plagiarized ones & similarity threshold 70%.

	Structure based	Keyword based	
# of correct guesses	7	4	Higher is better
# of incorrect guesses	2	7	Lower is better
# of gross incorrect guesses	0	4	Lower is better

Note: “gross incorrect guess” is give high similarity for two solutions that belongs to two different problems

Increase the threshold will decrease the number of correct guesses, but also decreases the number of incorrect guesses. Each algorithm works better on different data set. We still need more experiments to decide which algorithm better at given data set and threshold and what to improve on the system. But in general both algorithm are very good at detecting code plagiarism, according to experiment Structure based is better than keyword-based.

5.2. Functional experiment

In this experiment, we will perform a comparison between our implementation vs Jplag. the goal of this experiment is to test most common techniques used in source code plagiarism. and to find which system is better and if it immune to all techniques . The most common techniques used are:

1. Add and remove comments.
2. Rename identifiers.
3. Changing data types.
4. Reorder sequantial code
5. Reorder functional code.
6. Adding redundant statements.
7. Changing the structure of iteration and selection statements

	Jplag	OUR
1	PASS	PASS
In both algorithms comments are not consederd; so adding or delete them will not effect the results		
2	PASS	PASS
In both algorithms identifiers' names are not consederd; so renaming them will not has an effect		
3	FAIL	PASS
In Jplag changing data types will affect it's result, if there is a lot of changes the detection will fail In OUR changing data types will not affect it's result.		
4	FAIL/PASS	FAIL/PASS
In both algorithms the results aproximitly the same In OUR reslut is slitly better than Jplag In general reorder the sequantial code is not easy and there limit to do it		
5	FAIL/PASS	FAIL/PASS
In both algorithms the results aproximitly the same In Jplag result is slitly better than OUR Reorder functions decleration in functional code is easy, but the call of these functions will be sequantial wich helps OUR to detect some good similarity		
6	FAIL/PASS	FAIL/PASS
In OUR reslut is better than Jplag Adding redundant statements bettwen original code statements		
7	FAIL/PASS	FAIL/PASS
In OUR reslut is better than Jplag		

6. CONCLUSION

Hybrid algorithm proved to be much more accurate from the experiments that we conduct, And the system we build turned out be better than . We need to work some rare cases and involve more machine learning

7. REFERENCES

[1] A.Jadalla and A.Elnagar“PDE4Java: Plagiarism Detection Engine For Java Source Code: A Clustering Approach ”, 2007.

[2] L. Prechelt, M. Guido and M. Phlippsen, “JPlag: Finding plagiarisms among a set of programs”, Journal of Universal Computer Science, vol. 8, no. 11, 2000.

[3] L.Mousslades and A. Vakali “PDetect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets “, 2000.

[4] W.Chen, C.Duan, L.Zheng and Y.Zhao “A Structure based Method for Detecting Source-code Plagiarism in Computer Programming”,The European Conference on Education ,2013.

[5] Pycparcer, Retrieved from (pypi.python.org/pypi/pycparser)

[6] Clang, LLVM, Retrieved from (clang.llvm.org)