# SOFTWARE PLAGIARISM DETECTION

Aniruddha Wathare[1], Rohit Patil[2], Pooja Patil[3], Kajal Thapa[4]

[1] *Student, Computer Engineering., AISSMS's IOIT, Maharashtra, India*
[2] *Student, Computer Engineering., AISSMS's IOIT, Maharashtra, India*
[3] *Student, Computer Engineering., AISSMS's IOIT, Maharashtra, India*
[4] *Student, Computer Engineering., AISSMS's IOIT, Maharashtra, India*

## ABSTRACT

*With the development of internet and electronic devices, software plagiarism has become prevalent in software industries as well as educational institutes, violating one's intellectual integrity. Certain techniques such as watermarking and semantics-preserving code obfuscations were introduced to tackle this issue. However, besides the need to insert additional data in the original program, code obfuscations can often destroy watermarks. Also, it was found that a sufficiently determined attacker may be able to destroy any watermark. In order to overcome these issues, birth-marking technique is proposed, which extracts a set of characteristics that uniquely identify the original program. Our work focuses on extracting birthmarks from source codes, implementing algorithms to measure the similarity between them and displaying the results on a dedicated user interface with respect to a pre-set threshold.*

**Keyword** - *Plagiarism, Birth-marking, Jaccard, Cosine, Dice, Reflection, features, serialization.*

## 1. Introduction

Software Plagiarism is the practice of claiming, or implying, original authorship, or incorporating the code from someone else's source code in whole or in part, into one's own, without adequate acknowledgement.

Software Plagiarism is of major concern for the software industry as well as the Academia. Finding ways to identify and combat Plagiarism is central to maintaining one's intellectual integrity, and therefore a tool to detect Software Plagiarism is required. There are different tools that detect software plagiarism using different techniques like tokenization, parameterized-matching etc. but the idea was to conquer the inefficiencies of those techniques. Our work focuses on extracting birthmarks from source codes using Reflect API, implementing algorithms to measure the similarity between them and displaying the results on a dedicated user interface with respect to a pre-set threshold.

### 1.1 Background

#### Serialization

Serialization in java is a mechanism in which state of an object is written into a byte stream. In this process, data structures or object state can be transformed into a format that can be stored in file or memory buffer or transmitted and reconstructed later in a different or same computer environment.

#### Reflection API

Reflection is a part of java framework that provides a computer program an ability to examine, introspect, and modify its own structure and behavior at runtime [12].

Reflection can be used for observing and modifying program execution at runtime. A reflection-oriented program component can monitor the execution of an enclosure of code and can modify itself according to a desired goal related to that enclosure. This is typically accomplished by dynamically assigning program code at runtime [12].

In object oriented programming languages such as Java, reflection allows *inspection* of classes, interfaces, fields and methods at runtime without knowing the names of the interfaces, fields, methods at compile time. It also allows *instantiation* of new objects and *invocation* of methods [12].

**Software Metrics:**

There are several algorithms which are based on the software metrics. These algorithms extract several software metric features from a program and use this set of measures or features to compare programs for plagiarism. [11]

**Software Birthmarks**

Software Birthmarks are the inherent characteristics of the program. These are unique characteristics can be used to track or identify software thefts. We have extracted these birthmarks of the softwares to detect plagiarism. [1]

**Algorithms used:**

The algorithms et al. [4] [8] that we have used in the project are:

1. Cosine similarity
2. Jaccard similarity
3. Dice Coefficient

### 1.2 Methodology

**Reflection API** can be used to extract the unique characteristics of the program and change or modify these dynamic characteristics at run time.

We have used reflection API to extract:

1. Number of Functions.
2. Number of Constructors.
3. Number of Variables.
4. Number of Parameters.
5. Class Name.
6. Return type of functions.
7. Type of Variables.
8. Name of the Parent Class

Certain Software metrics of the software code have been extracted in order to compare with those of the other programs to detect plagiarism. The software metrics include:

1. Number of for loops
2. Number of if conditions
3. Number of else keywords
4. Number of try and catch keywords
5. Number of do-while loops

6. Number of while loops

We have used **serialization** to store these extracted feature sets of the codes into a file. Java provides automatic serialization which requires that the object be marked by implementing the **java.io.Serializable** interface.

The code to be checked for plagiarism has its feature set extracted and compared with those that are present in the database. To check code with maximum similarity we have used 3 algorithms:

1. Jaccard Index: is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

2. Cosine similarity: According to [12] The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$similarity = cos\theta = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

3. Dice coefficient: According to [12] the set operations can be expressed in terms of vector operations over binary vectors $A$ and $B$:

$$s_v = \frac{2\,|A.B|}{|A|^2 + |B|^2}$$

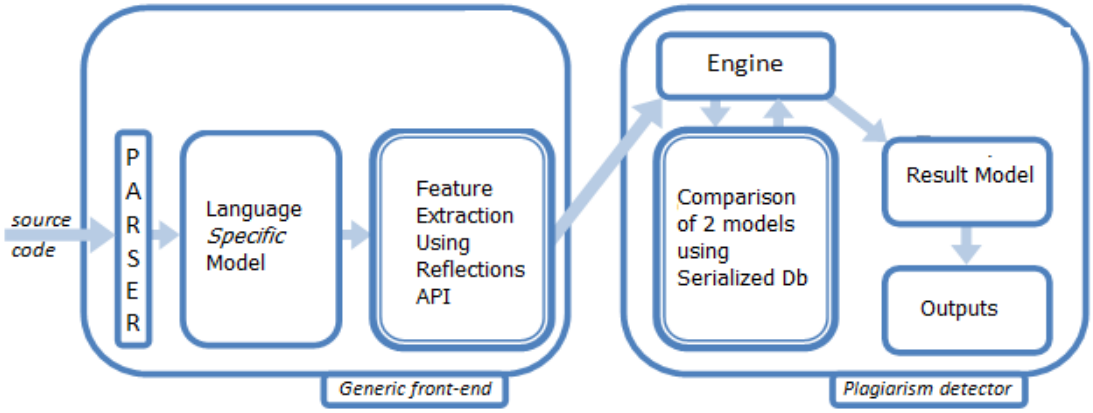
which gives the same outcome over binary vectors and also gives a more general similarity metric over vectors in general terms.

## 2. Proposed System

In the project we propose a system in which, initially we extract the basic characteristics of the programs using simple keyword matching, such as number of different loops, saving them in separate variables and serializing them as a part of the feature set into the database.

Further, arraylist is used to store the other aforementioned extracted characteristics into the database such as,

1. Number of Functions.
2. Number of Constructors.
3. Number of Variables.
4. Number of Parameters.
5. Class Name.
6. Return type of functions.
7. Type of Variables.
8. Name of the Parent Class




These are collectively maintained in the database as feature sets, to make comparisons with the feature set of the source code to be checked for plagiarism.

ARCHITECTURE DIAGRAM

The feature set of the source code to be checked for plagiarism is then extracted and checked against those in the database using the similarity algorithms jaccard, cosine and dice. Following block diagram gives a clear view;

## 3. Algorithms

The algorithms that we have used in our project are as follows:

### 3.1 Jaccard Index

Let c be the set of codes where

Extract all code

for i in range 1 to c

codefeature = extractfeature(i)

current_codefeature = extractcurrent_code($C_i$)

current_val = compare_code($C_i$, C)

end for loop

sort_code according to comparison method

find max_match with code

Apply jaccard formula

$$j(a,b) = \frac{C_i \,.\, C}{|C_i|^2 + |C|^2 - C_i \,.\, C}$$

Generate final result

**3.2 Cosine similarity**

Let C be the set of codes where

extract all code

for i in range 1 to c

codefeature=extractfeature(i);

current_codefeature=extractfeature(C);

current_val=compare_code($C_i$,C);

end for loop

sort code according to comparison method

find maximum match with the code

compare_code($C_i$,C)

cosine formulae :

$$\alpha = \cos^{-1}\left(\frac{C_i \cdot C}{|C_i| . C}\right)$$

return value

generate final results.

**3.3 Dice Coefficient**

Let c be the set of codes where

Extract all code

for i in range 1 to c

codefeature = extractfeature(i)

current_codefeature = extractcurrent_code($C_i$)

current_val = compare_code($C_i$, C)

end for loop

sort_code according to comparison method

find max_match with code

Apply Dice formula

$$s_v = \frac{2|C_i . C|}{|C_i|^2 + |C|^2}$$

Generate final result

## 4. Experimental Results

To test the results of the proposed systema program named seats.java was checked for plagiarism, its feature sets were extracted and compared with those in the database.

The code was checked for plagiarism using all the 3 aforementioned algorithms separately as well as collectively, and the features of the source code with which it had maximum similarity (ticketing.java) was displayed.

Following are the features of the source code to be checked for plagiarism (seats.java):

| | |
|---|---|
| Class Name | Seats |
| Parent Class Name | javax.swing.JDialog |
| Access Specifier of Class | public |
| Number of methods | 0 |
| Number of fields | 1 |
| Number of Constructors | 1 |
| Number of for loops | 3 |
| Number of if loops | 0 |
| Number of else loops | 1 |
| Number of do loops | 0 |
| Number of while loops | 0 |
| Number of try loops | 2 |
| FIELDS WITH RETURN TYPE : | |
| int | 0 |
| float | 0 |
| double | 0 |
| long | 0 |
| short | 0 |
| string | 0 |
| METHODS WITH RETURN TYPE : | |
| int | 0 |
| float | 0 |
| double | 0 |
| long | 0 |
| short | 0 |
| string | 0 |
| void | 0 |

The above features were found to have maximum similarity with the features of ticketing.java whose features are the following:

| | |
|---|---|
| Class Name | Ticketing |
| Parent Class Name | java.lang.Object |
| Access Specifier of Class | public |
| Number of methods | 3 |
| Number of fields | 2 |
| Number of Constructors | 1 |
| Number of for loops | 0 |
| Number of if loops | 0 |
| Number of else loops | 0 |
| Number of do loops | 0 |
| Number of while loops | 0 |
| Number of try loops | 1 |
| FIELDS WITH RETURN TYPE : | |
| int | 0 |
| float | 0 |
| double | 0 |
| long | 0 |
| short | 0 |
| string | 0 |
| METHODS WITH RETURN TYPE : | |
| int | 0 |
| float | 0 |
| double | 0 |
| long | 0 |
| short | 0 |
| string | 0 |
| void | 2 |

**Results**

The results of comparison of the above feature sets were:

**Jaccard : 30%**
**Cosine : 46.16%**
**Dice : 46.15%**
**All : 40.77%**

**4.1 Venn Diagram**

We have used a certain pre-set threshold to cast the source codes as unique or plagiarized.

The pre-set threshold that we have set in our project is **50%.**
A source code will be cast as plagiarized code if the plagiarism percentage obtained is above 50% else it is a unique code.

## 5. CONCLUSIONS

With the growing trend of illegal code reuse, software thefts are becoming more common, compromising one's intellectual integrity. The proposed systemcan be used to settle the disputes of software thefts in software industries as well as in academic institutes to check the uniqueness of the software projects.

Mainly, the systemcomputes and displays the similarity value between the source code of the software to be checked for plagiarism and those in the database. Compared to conventional systems that are used to check for plagiarism, this systemproposes a different view of plagiarism detection by using utilities of java framework to extract and compare the feature sets.

## 6. REFERENCES

[1] Software Plagiarism Detection with Birthmarks based on Dynamic Key Instruction Sequences Zhenzhou Tian, Qinghua Zheng, Member, IEEE, Ting Liu, Member, IEEE, Ming Fan, Eryue Zhuang and Zijiang Yang, Senior Member, IEEE.

[2] Zhenzhou Tian, Qinghua Zheng, Ting Liu, Ming Fan, Eryue Zhuang and Zijiang Yang, "Software Plagiarism Detection with Birthmarks based on Dynamic Key Instruction Sequences", 2015 IEEE Transactions.

[3] Yoon-Chan Jhi, Xinran Wang, Xiaoqi Jia, Sencun Zhu, Member, Peng Liu and Dinghao Wu, "Program Characterization Using Runtime Values and Its Application to Software Plagiarism Detection" ,IEEE TRANSACTIONSON SOFTWARE ENGINEERING, 2015

[4] Fangfang Zhang, Dinghao Wu, Peng Liu and Sencun Zhu, "Program Logic Based Software Plagiarism Detection", 2014 IEEE 25th International Symposium on Software Reliability Engineering.

[5] Shanmugasundaram Hariharan, "Automatic Plagiarism Detection Using Similarity Analysis", The International Arab Journal of Information Technology, Vol. 9, No. 4, July 2012.

[6] Samer Abd El -Wahed, Ahmed Elfatatry and Mohamed S. Abougabal, "A New Look at Software Plagiarism Investigation and Copyright Infringement", 2007 IEEE.

[7] Tapan P. Gondaliya, Hiren D. Joshi and Hardik Joshi, "Source Code Plagiarism Detection SCPDet: A Review", International Journal of Computer Applications.

[8] Vikas Thada and Dr Vivek Jaglan ,"Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm", International Journal of Innovations in Engineering and Technology (IJIET).

[9] Georgina Cosma and Mike Joy, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis", IEEE TRANSACTIONS ON COMPUT-ERS, VOL. 61, NO. 3, MARCH 2012

[10] Asako Ohno and Hajime Murano, "A TWO-STEP IN-CLASS SOURCE CODE PLAGIARISM DETECTION METHOD UTILIZING IMPROVED CM ALGORITHM AND SIM", International Journal of Innovative Computing, Information and Control ICIC International Volume 7, Number 8, August 2011.

[11] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn and Supachanun Wanapu, "Using of Jaccard Coefficient for Keywords Similarity", Proceedings of the International Multi Conference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong.

[12] www.wikipedia.com