PROJECT TITLE

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

by

Deven Bhalerao

Roll No. 05

Supervisor:

Prof. Prasad Padalkar

Department of Information Technology

Don Bosco Institute of Technology

2015-2016

AFFILIATED TO

UNIVERSITY OF MUMBAI

DON BOSCO INSTITUTE OF TECHNOLOGY
Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled **"SoftPlag"** is a bonafide work of

Deven Bhalerao	**05**
Manish Jain	**28**
Vishal Jha	**29**
Kunal Naik	**45**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Undergraduate** in **Bachelor of Information Technology**

Date: / /

Prof. Prasad Padalkar
(Supervisor)

Prof. Janhavi Baikerikar
(HOD, IT Department)

Dr. Prasanna Nambiar
(Principal)

DON BOSCO INSTITUTE OF TECHNOLOGY
Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

Project Report Approval for B.E.

This project report entitled **”SoftPlag”** by **Deven Bhalerao, Manish Jain, Vishal Jha, Kunal Naik** is approved for the degree of **Bachelor of Engineering in Information Technology**

(Examiner’s Name and Signature)

1. ____________________________

2. ____________________________

(Supervisor’s Name and Signature)

1. ____________________________

(Chairman)

1. ____________________________

Date:

Place:

DON BOSCO INSTITUTE OF TECHNOLOGY

Vidyavihar Station Road, Mumbai - 400070

Department of Information Technology

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(——————————-)

(Deven Bhalerao)

Date:

ABSTRACT

The purpose of this project is to detect plagiarism of source code in the files given by user. This plagiarism detection software will be able to detect high level plagiarism which involves minor changes in logic, changes in flow of program etc. as well as low level plagiarism which involves changing variable names, adding comments etc.

Plagiarism is widespread practice in both the software industry and the educational institutes. Companies face a constant risk of getting their code stolen by competitors and incurring a huge economic loss. Students in colleges and schools also indulge in plagiarism in their programming assignments . Plagiarism can be done easily and this makes a plagiarism detection software an extremely useful tool.

Keywords: Plagiarism, Java

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

To develop a tool for detecting plagiarism in software source code using the machine learning algorithms.

1.2 Scope of the Project

- For the first level of implementation , tool will be on a local machine for checking plagiarism.
- The source and target program should be in Java.

1.3 Current Scenario

1.4 Need for the Proposed System

The Role of Plagiarism detection in Education, Role of Plagiarism Detection in Software,Industry especially copyright of software.

	MOSS	SHERLOCK	JPLAG	CODEMATCH
Cost	Free	Open Source	Free	It is a Commercial T
Safety	Sign-in Required	Executes at Local machine	Sign-in Required	Executes at Local m
Service	Internet	Standalone	Web-Service	Standalone
Algorithm	Winowing	Token Matching	Greedy String Tiling	String Matching
Speed	Fast	More files requires more time	Fast	More files requires r

1.5 Summary of the Results / Task completed

Study of different Research Papers,Thesis on Source Code Plagiarism. Survey on different source code plagiarism tools. Implementation upto Level 3 Plagiarism (Rule-based).

Chapter 2

Review of Literature

2.1 Summary of the investigation in the published papers

- **Computer Algorithms for Plagiarism Detection**

  Described the various levels of plagiarism and the different metrics(approaches) like Halstead metric, ACCUSE metric, FORTRAN programs, etc. This paper also compared different algorithms based on the software metrics used. The different levels of plagiarism defined in this paper will form the foundation of our project and we'll be looking to detect plagiarism based on these levels.

- **A Comparison of Similarity Techniques for Detecting Source Code Plagiarism**

  This paper outlines different modern approaches to software similarity measurement. Algorithms like Levenshtein edit distance, Tree edit distance and graph edit distance were discussed. Knowledge of different approaches was gained and this will help us choose our approach.

- **An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis**

  Latent Semantic Analysis, a statistical approach to detecting similarity is analysed in-depth in this paper. This paper also has an exhaustive description of what constitutes Plagiarism which includes surveys and research papers. Deeper understanding of Plagiarism was made possible by this research paper.

- **Plagiarism Detection in Java Code**

  This thesis gives a step-by-step procedure on how to detect plagiarism in java source code using the Levenshtein Edit distance algorithm. It uses different normalization techniques and demonstrates these techniques using examples. Normalization techniques shown inn this thesis will be used in our project.

- **Source Code Plagiarism Detection ‘SCPDet’: A Review**

  In this paper author describes the real meaning of source code plagiarism and then described the different source code plagiarism detection tools and compared its function, characteristics and technique. In the last phase, authors discussed the different research papers and compared in tabular form with its technique, method, characteristics, functionality and its result.

2.2 Comparison between the tools / methods / algorithms

Table 2.1: Compression of four source code detection tools with its characteristics, function and technique

Tools	JPlag	SIM	MOSS	Plaggie
Open Source Tools/Paid	NO	YES	NO	YES
Local/online tool	Web	Local	Web	Local
Code Submit/File	Submit Code	Submit File	Submit Code	Submit Code
Lang. Support	6	5	23	1
Expandability	No	Yes	No	No
Founded in Year	1996	1989	1994	2002
Founded By	Guido Malpohl	Dick Grune	Aiken	Ahtiaine
Technique	Greedy String Tiling and Optimization and Tokenization	Flax lexical analyzer	Winnowing technique	Greedy String Tiling and Tokenization

Table 2.2: Comparison between different metrics : Structural Metrics and Similarity Metrics

Structural metrics	Similarity metrics
Structural metrics – no. of variables, no. of keywords, no. of loops, no. of comment lines	Similarity metrics – no. of characters per line, no of code lines, no. of blank lines
Structural metrics represent information about programming constructs and elements used in the code.	Similarity metrics are indicative of the style used in programming and are effective in detecting plagiarism.
Lots of rudimentary plagiarism detection algorithms like Halstead use only structural metrics and are ineffective for larger programs.	Modern approaches include algorithms that use a combination of structural and similarity metrics to detect plagiarism and are highly effective.

2.3 Algorithms

- **Level 0 Plagisrism Pseudo-Code**

```
Read contents of both files
While all lines of file 1 and file 2 have not been visited
        if lines in file 1 and file 2 are not perfect match
                print “found difference”
                break out of loop
if all lines are perfect match
        print plagiarised
```

- **Level 1 Plagisrism Pseudo-Code**

```
Read contents of both files
While all lines of file 1 and file 2 have not been visited
        While current lines in file 1 & file 2 are not code lines
                If line starts with “//” or is empty
                        Skip line
                If line starts with /*
                        Skip all lines until */ is detected
                If line is code line
                        set iterator condition as fulfilled and end loop
        if lines in file 1 and file 2 are not perfect match
                print “found difference” and break out of loop
if all lines are perfect match
        print plagiarised
```

- **Level 3 Plagisrism Pseudo-Code**

```
Read contents of both files
While all lines of file 1 and file 2 have not been visited
    While current lines in file 1 & file 2 are not code lines
        If line starts with “//” or is empty
            Skip line
        If line starts with /*
            Skip all lines until */ is detected
        If line is code line
            count number of variables in code line and end loop
If number of variables in both files are equal
    print plagiarised
```

Chapter 3

Analysis and Design

3.1 Methodology / Procedure adopted

- The common ways adopted by people for plagiarizing the code are:
  1. The original source code can be replicated as it is.
  2. Addition of comments in the source code.
  3. Modification in identifiers.
  4. Chane in variable position.
  5. The procedure combination can be done in source code.
  6. Program statements can be changed by some modifications.
  7. Control Logic can be modified.
- Describe on the development methodology / model you would use. (E.g. Agile method or Iterative Model)
  1. In Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.
  2. An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

- How you intend manage the weekly meetings ?
  The weekly meetings need to managed properly because in order to accomplish the goals desired, you will need to have a good strategic and tactical plan. In the meeting, plans may be decided by each team member and the procedure is been planned.

- How do you intend to monitor and measure the progress of the project?
  The monitoring and measure of progress of report is been done on basis of different modules. A schedule is maintained for each module to be complemented. Github is used for project monitoring ,as all the team members upload their work on completion.

3.2 Analysis

Based on the requirements gathered, how was the feasibility study of the project carried out?
The project is about the detection of software source code plagiarism ,so the study carried out on the comparison of the different plagiarism software's which are based on rule based algorithms.
On comparison of different features of some software's.
The papers were referred for plagiarism detection, which gives the idea of different levels of plagiarism and defining the metrics for the programs.
If any requirements, were modified why they were modified?

3.2.1 Software / System Requirement Specification - IEEE format

3.3 Proposed System

Give the details of your proposed system and architecture Advantage of the proposed system over the existing system

3.3.1 Hardware / Software requirements

Development Hardware / Software requirements
Deployment Hardware / Software requirements

3.3.2 Design Details

Different UML diagrams as per the project requirement (For e.g. Use Case Diagram)

3.3.3 Implementation Plan

Timeline chart is for Next semester

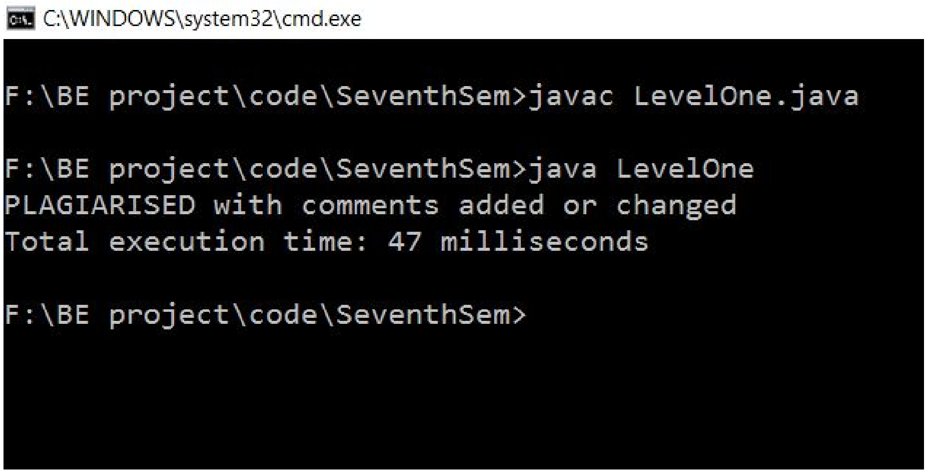
Chapter 4

Results and Discussion

4.1 Result of Comparison for various levels for Execution time

Two files were given as input and one of these files is plagiarised and the other is source code i.e original.

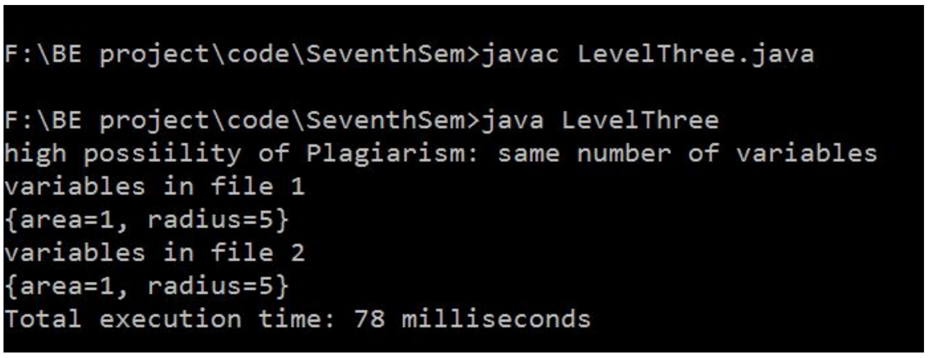
- Level One Plagiarism Detection.



```
C:\WINDOWS\system32\cmd.exe

F:\BE project\code\SeventhSem>javac LevelOne.java

F:\BE project\code\SeventhSem>java LevelOne
PLAGIARISED with comments added or changed
Total execution time: 47 milliseconds

F:\BE project\code\SeventhSem>
```

The plagiarised has comments and extra whitespace added.The program correctly identifies the that the file has been plagiarised.

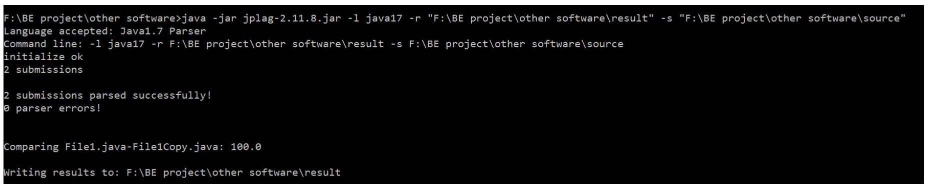
- Level Three Plagiarism Detection.



```
F:\BE project\code\SeventhSem>javac LevelThree.java

F:\BE project\code\SeventhSem>java LevelThree
high possiility of Plagiarism: same number of variables
variables in file 1
{area=1, radius=5}
variables in file 2
{area=1, radius=5}
Total execution time: 78 milliseconds
```

The plagiarised has changed the names of the variables and shifted their positions around the file.The program correctly identifies the that the file has been plagiarised.

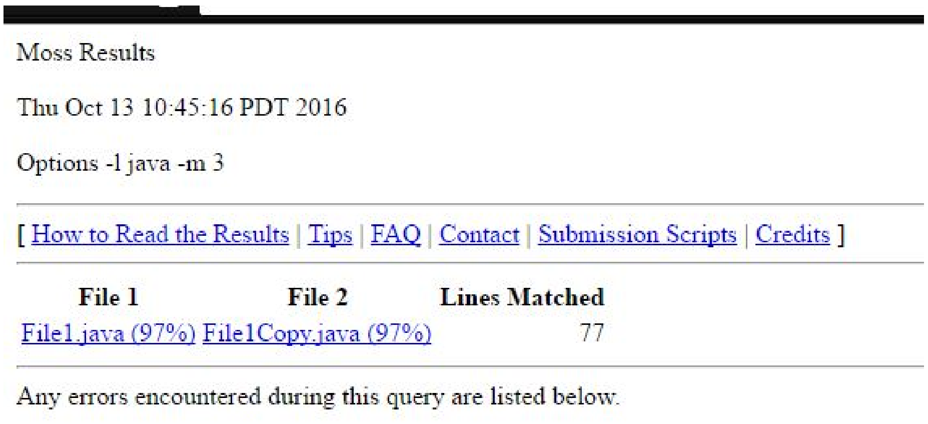
- Result from JPlag.



```
F:\BE project\other software>java -jar jplag-2.11.8.jar -l java17 -r "F:\BE project\other software\result" -s "F:\BE project\other software\source"
Language accepted: Java1.7 Parser
Command line: -l java17 -r F:\BE project\other software\result -s F:\BE project\other software\source
initialize ok
2 submissions

2 submissions parsed successfully!
0 parser errors!


Comparing File1.java-File1Copy.java: 100.0

Writing results to: F:\BE project\other software\result
```

These two files were given as an input to the popular plagiarism detction software tool called JPlag and the results shown by this tool matched the result given by our program.

- Result from MOSS.



Moss Results

Thu Oct 13 10:45:16 PDT 2016

Options -l java -m 3

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

File 1	File 2	Lines Matched
File1.java (97%)	File1Copy.java (97%)	77

Any errors encountered during this query are listed below.

These two files were given as an input to the popular plagiarism detction software tool called MOSS and the results shown by this tool matched the result given by our program.

Chapter 5

Conclusion

SoftPlag is a useful tool to help detect plagiarism in source code.It is able to identify a wide range of plagiarism types which includes comment addition, changing names of variables and changing control flow of program.It can be used by both educational institutes as well as software companies to prevent plagiarism.

Appendix - I

Data Sheet(s) - Electronic component

Installation Procedure - Development Software

References

[1] Zhi Zhou, Member, IEEE, Gonzalo R. Arce, Fellow, IEEE, and Giovanni Di Crescenzo; *Halftone Visual Cryptography*; IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 15, NO. 8, AUGUST 2006

[2] HTML 5 `http://en.wikipedia.org/wiki/HTML5` , last modified on 6 October 2014

Acknowledgements

Parargraph 1 of you acknowledgement

Parargraph 2 of you acknowledgement

(————————————-)
(Signature)

(————————————-)
(Name of Student and Roll No.)

Date: