Merry: Web-based Code Clone Detection System using Machine Learning

MERRY: WEB-BASED CODE CLONE DETECTION SYSTEM USING MACHINE LEARNING

เมอร์รี่ ระบบตรวจจับโค้ดโคลนผ่านเว็บโดยใช้การเรียนรู้ของเครื่อง

BY

MR. VARA	**ARAMMONGKOLVICHAI**	**5988060**
MR. WEEKIT	**AUSAVASERENONT**	**5988067**
MISS. WANNAPORN	**VICHAISRI**	**5988266**

ADVISOR
DR. CHAIYONG RAGKHITWETSAGUL

CO-ADVISOR
DR. MORAKOT CHOETKIERTIKUL

A Senior Project Submitted in Partial Fullfillment of the Requirement for

THE DEGREE OF BACHELOR OF SCIENCE
(INFORMATION AND COMMUNICATION TECHNOLOGY)

Faculty of Information and Communication Technology
Mahidol University
2019

ACKNOWLEDGEMENTS

We would like to give gratitude to everyone who has been involved with the development and the success of this project and thank them for their assistance throughout the entire period of this project. Dr. Chaiyong Ragkhitwetsagul, our senior project advisor, Dr. Morakot Choetkiertikul, our Co-advisor. We would also like to thank the Faculty of Information and Communication Technology, instructors, staff, members for the support given to us. Lastly, we would also like to thank our families for giving us support while making this project. The project would not have been this successful without those mentioned and their support that was offered.

Mr. Vara Arammongkolvichai
Mr. Weekit Ausavaserenont
Miss. Wannaporn Vichaisri

MERRY: WEB-BASED CODE CLONE DETECTION SYSTEM USING MACHINE LEARNING

MR. VARA	ARAMMONGKOLVICHAI	5988060 ITCS/B
MR. WEEKIT	AUSAVASERENONT	5988067 ITCS/B
MISS. WANNAPORN	VICHAISRI	5988266 ITCS/B

B.Sc.(INFORMATION AND COMMUNICATION TECHNOLOGY)

PROJECT ADVISOR: DR. CHAIYONG RAGKHITWETSAGUL

ABSTRACT

Code clones (similar fragments of code) generally occur in software project due to intentional copying and pasting of code or unintentional independently developed code. Code clones can be found by using automated tools called code clone detection tools. However, the existing tools still face challenges when detecting clones with several modifications. This is partially because the existing techniques, for instance textual approach, lexical approach, syntactic approach, and semantic approach, still rely on rigid code matching techniques or only compare structure of the code.

To avoid using rigid pattern matching rules of the traditional clones detection techniques, we can use machine learning, that detects clones based on existing cloned data, to detect clones instead. We investigate the effectiveness of using machine learning as a code clone detection techniques in this project. Moreover,the existing clones detection tools mostly have to run via command line. Thus, this project proposes a clone detection system that runs as a web application to provide a better user experience.

We build the web-based clone detection tool, called Merry, by using 4 machine learning models including decision tree (REPTree), random forest, support vector machine (SVM), and Support Vector Machine using Sequential Minimal Optimization (SVM using SMO). The evaluation of our clone detection performance using the BigCloneBench, which is the largest clone ground truth database, shows that our tool has high precision and recall. In addition, the user study of Merry web application also demonstrates that the tool is easier to use and friendlier than the well-known command line based tool, Simian.

KEYWORDS: SOFTWARE ENGINEERING, CODE CLONE DETECTION, MACHINE LEARNING

71 P.

เมอร์รี่ ระบบตรวจจับโค้ดโคลนผ่านเว็บโดยใช้การเรียนรู้ของเครื่อง

นาย วรา	อร่ามมงคลวิชัย	5988060 ITCS/B
นาย วีกิจ	อัศวเสรีนนท์	5988067 ITCS/B
นางสาว วรรณพร	วิชัยศรี	5988266 ITCS/B

วท.บ. (เทคโนโลยีสารสนเทศและการสื่อสาร)

อาจารย์ที่ปรึกษาโครงการ: ดร. ชัยยงค์ รักขิตเวชสกุล

บทคัดย่อ

โค้ดโคลน (ส่วนของโค้ดที่เหมือนกัน) มักจะเกิดขึ้นในโครงการซอฟต์แวร์ เนื่องจากการตั้งใจคัดลอกโค้ดหรือการพัฒนาโค้ดอย่างอิสระโดยไม่ได้ตั้งใจ โค้ดโคลนสามารถพบได้โดยใช้เครื่องมืออัตโนมัติที่เรียกว่าเครื่องมือตรวจจับโคลน อย่างไรก็ตาม เครื่องมือตรวจจับโคลนในปัจจุบันยังคงประสบปัญหาเมื่อต้องจรวจจับโคลนที่มีการแก้ไขจำนวนมาก เหตุผลบางส่วนเนื่องจากเทคนิคที่มีอยู่ เช่น วิธีการตรวจจับแบบข้อความ, วิธีการตรวจจับแบบโดยใช้คำ, วิธีการตรวจจับโดยใช้โครงสร้างภาษา และวิธีการการตรวจจับเชิงความหมาย ยังคงพึ่งพาเทคนิคการจับคู่โค้ดที่ตายตัวหรือเพียงแค่เปรียบเทียบโครงสร้างของโค้ดเท่านั้น

เพื่อหลีกเลี่ยงวิธีการแบบดั้งเดิมที่มีการจับคู่โค้ดแบบมีรูปแบบตายตัวเพื่อตรวจจับโคลน พวกเราเลือกใช้การเรียนรู้ของเครื่องเพื่อตรวจจับโคลน เนื่องจากวิธีการนี้ใช้ข้อมูลของโคลนที่มีอยู่เพื่อสร้างเครื่องมือตรวจจับโคลนแทนที่จะใช้รูปแบบตายตัวดังกล่าวมาข้างต้น ดังนั้นในโครงการนี้พวกเราศึกษาประสิทธิผลของการใช้การเรียนรู้ของเครื่องเพื่อตรวจจับโคลน นอกจากนี้เครื่องมือการตรวจจับโคลนที่มีอยู่ในปัจจุบัน ส่วนมากต้องรันโดยใช้บรรทัดคำสั่ง ดังนั้น โปรเจคนี้ยังเสนอระบบตรวจจับโคลนซึ่งรันบนเว็บแอปพลิเคชั่น เพื่อมอบประสบการณ์การใช้งานที่ดีขึ้น

พวกเราสร้างเครื่องมือตรวจจับโคลนใช้งานผ่านเว็บที่มีชื่อว่า เมอร์รี่ โดยใช้โมเดลการเรียนรู้ของเครื่อง 4 แบบ ประกอบด้วย แผนภาพต้นไม้, แรนด้อมฟอเรส, ซัพพอร์ตเวกเตอร์แมชชีน และซัพพอร์ตเวกเตอร์แมชชีนร่วมกับซีเควนเชี่ยลมินิมอลออปติไมซ์เซชั่น การประเมินประสิทธิภาพของเครื่องมือตรวจจับโคลน พวกเราใช้ บิ๊กโคลนเบ็นซ์ ซึ่งเป็นฐานข้อมูลโคลนที่เชื่อถือได้ที่สุด ผลจากการประเมินประสิทธิภาพแสดงให้เห็นว่าเครื่องมือของพวกเรามีความแม่นยำสูง นอกจากนี้การศึกษาผู้ใช้ของเมอร์รี่เว็บแอปพลิเคชั่นยังแสดงให้เห็นว่าเครื่องมือนี้ใช้งานง่ายและเป็นมิตรกว่าเครื่องมือตรวจจับโคลนที่รู้จักกันดีชื่อ ซิเมียน ที่เป็นเครื่องมือแบบบรรทัดคำ

71 หน้า

CONTENTS

Page

ACKNOWLEDGMENTS ii
ABSTRACT iii
LIST OF TABLES viii
LIST OF FIGURES ix
LIST OF LISTINGS xi

1 INTRODUCTION 1
1.1 MOTIVATION 1
1.2 PROBLEM STATEMENTS 2
1.3 OBJECTIVES OF PROJECT 2
1.4 SCOPE OF THE PROJECT 3
1.5 EXPECTED BENEFITS 3
1.6 ORGANIZATION OF THE DOCUMENT 3
2 BACKGROUND 5
2.1 FUNDAMENTALS 5
2.1.1 CODE CLONES 5
2.1.2 MACHINE LEARNING 9
2.1.3 WEB APPLICATION 10
2.2 TOOLS AND TECHNIQUES 10
2.2.1 GITHUB 10
2.2.2 GITHUB API 11
2.2.3 NODE.JS 11
2.2.4 MONGODB 12
2.2.5 WEKA 12
2.2.6 JAVA PARSER 12
2.2.7 JAVA TOKENIZER 12
2.2.8 CODE2VEC 13

2.3 LITERATURE REVIEW 15

2.3.1 SYNTACTIC CODE CLONE DETECTION TECHNIQUE 15

2.3.2 USING MACHINE LEARNING TO IMPROVE EXISTING CODE CLONE DETECTION TECHNIQUE 17

2.3.3 USING MACHINE LEARNING FOR CODE CLONE DETECTION 18

2.3.4 DATASET FOR EVALUATING CODE CLONE DETECTION TECHNIQUES 19

2.4 COMPARISON TO RELATED WORK 20

3 ANALYSIS AND DESIGN 22

3.1 SYSTEM ARCHITECTURE OVERVIEW 22

3.2 USE CASE ANALYSIS 23

3.3 STRUCTURE CHART 24

3.4 SYSTEM ANALYSIS 26

3.4.1 CONTEXT DIAGRAM 26

3.4.2 DATA FLOW DIAGRAM LEVEL 1 26

3.4.3 DATA FLOW DIAGRAM LEVEL 2 26

4 IMPLEMENTATION 32

4.1 BUILDING ML CODE CLONE DETECTION ENGINE 32

4.1.1 DATA COLLECTION AND PREPARATION 32

4.1.2 CODE METRICS EXTRACTION 36

4.1.3 ML MODEL SELECTION 37

4.2 USING TRAINED ML MODELS FOR CLONE DETECTION 40

4.3 BUILDING A WEB APPLICATION 41

4.3.1 FRONT-END 41

4.3.2 BACK-END 42

5 EVALUATION RESULTS 49

5.1 EVALUATION OF MERRY CLONE DETECTION ENGINE USING BCB 49

5.1.1 METHODOLOGY 49

5.1.2 ERROR MEASURES 49

5.1.3 EVALUATION RESULTS.. 51
5.1.4 DISCUSSION 54
5.2 EVALUATION OF MERRY CLONE DETECTION ENGINE ON REAL SOFTWARE PROJECTS.. 54
5.2.1 METHODOLOGY 54
5.2.2 EVALUATION RESULT 54
5.3 EVALUATION OF MERRY WEB APPLICATION BY USERS....... 57
5.3.1 METHODOLOGY 57
5.3.2 EVALUATION RESULTS.. 58
6 CONCLUSIONS.. 63
6.1 CONCLUSION.. 63
6.2 PROBLEMS AND LIMITATIONS 63
6.3 FUTURE WORK 64
REFERENCES 65
BIOGRAPHIES 70
APPENDIX A 71

LIST OF TABLES

Page

Table 2.1: Comparison with Existing Tools 21
Table 4.1: BigCloneBench Data Summary 35
Table 4.2: Amount of Training Data 35
Table 4.3: Amount of Testing Data 35
Table 4.4: Syntactic Code Metrics 37
Table 5.1: Merry execution parameters 49
Table 5.2: Experimental result on sampled BCB dataset 52
Table 5.3: Merry's result on real software project 55

LIST OF FIGURES

Page

Figure 2.1: Set of AST path of finding method 14
Figure 2.2: Generalized Suffix Tree (GST) 16
Figure 2.3: Overview Diagram of Oreo's Clones Detection Pipeline........ 17
Figure 3.1: System Architecture 22
Figure 3.2: Use case Diagram 23
Figure 3.3: Structure Chart........ 25
Figure 3.4: Context Diagram 28
Figure 3.5: Data Flow Diagram Level 1 29
Figure 3.6: Prepare Data Process 29
Figure 3.7: Train Model Process........ 30
Figure 3.8: Detect Clones Process........ 31
Figure 4.1: Training process 32
Figure 4.2: BigCloneBench Structure 34
Figure 4.3: code2Vec Process 37
Figure 4.4: An Example of a Decision Tree [1]........ 38
Figure 4.5: Random Forest Simplified 39
Figure 4.6: How SVM classify the data [2] 40
Figure 4.7: Detection process........ 41
Figure 4.8: Structure of the two collections in the MongoDB database........ 45
Figure 4.9: Home page........ 45
Figure 4.10: Login page 46
Figure 4.11: Repo page 46
Figure 4.12: Result page........ 47
Figure 4.13: History page 48
Figure 4.14: Setting page........ 48
Figure 5.1: Terminology of the confusion matrix [3]........ 50
Figure 5.2: Precision and Recall measurement [3] 51

Figure 5.3: Detection process 52
Figure 5.4: Detection process 53
Figure 5.5: Detection process 53
Figure 5.6: Question 1 59
Figure 5.7: Question 2 60
Figure 5.8: Question 3 60
Figure 5.9: Question 4 61
Figure 5.10: Question 5 61
Figure 5.11: Question 6 62

LIST OF LISTINGS

Page

Listing 1.1: Java implementation of bubble sort algorithm 1
Listing 2.1: Java binary search source code 15
Listing 4.1: Connection with GitHub 42
Listing 4.2: Connection with GitHub 42
Listing 4.3: Write data to MongoDB 43
Listing 4.4: Read data from MongoDB 44
Listing 5.1: Example of challenge true clone 55
Listing 5.2: Example of false clone 56

CHAPTER 1 INTRODUCTION

1.1 Motivation

Code clones are similar segments of code. Most software systems and various open-source software contain duplicate parts of code or code clones that are generated by copying and pasting the same source code, which is the normal behavior of software developers. Roy et al.'s research [4] shows that about 7% to 23% of all over software system consists of clone fragments. Code clone detection is a trendy topic in the software engineering field. It is an approach to detect clones in software. These duplicated parts of code were a cause of code redundancy and several bugs when developers incompletely update all the duplicated code segments when they are making changes to the source code.

Presently, several techniques can be used to detect code clones, which are text-based, token-based, tree-based, and graph-based; however, the accuracy results are still low on a lot of modification. Machine Learning technique is a new technology that can predict the outcome by learning from past data. We are interested in investigating if it may enhance the performance of existing code clone detection tools.

Listing 1.1: Java implementation of bubble sort algorithm

```java
public void bubblesort(String filenames[]){
   for(int i = filenames.length-1; i > 0 ; i--){
      for(int j = 0 ; j < i ; j++){
        String temp;
        if(filename[j].compareTo(filenames[j + 1]) > 0){
        temp = filenames[j];
        filenames[j] = filenames[j + 1];
        filenames[j+1] = temp;
        }
      }
   }
}
```

```
private voidsortByName() {
  int i , j ;
  String v;
  for( i = 0; i < count; i++) {
     ChannelItem ch = chans[i ];
     v = ch.getTag() ;
     j = i ;
     while(( j > 0) && ( collator .compare(chans[-j1].getTag() , v) > 0) ) {
        chans[j] = chans[-j1];
        0--j;
              }
  chans[j] = ch;
  }
}
```

1.2 Problem Statements

This project tackles the following problems in code clone detection:

1. The most important and challenging task in code clone detection development is to make a tool that detect clones with high precision and high recall. The existing techniques and tools are still facing challenges when detecting clones with several modifications (e.g., added/deleted/modified statements).

2. Existing clone detection tools are difficult to use for naive programmers because they are mostly created as a command-line program with poor user interface and limited visualization and reporting techniques.

1.3 Objectives of project

The objectives of the project are as follows:

1. To create a code clone detection tool using machine learning techniques and study its effectiveness.

2. To enhance the user experience of code clone detection tools as follows:

   (1) providing code clone detection as a web application to users

   (2) providing visualization of clone results

1.4 Scope of the project

The project falls under the following scope:

- The system must support connection to GitHub account and able to clone a repository.
- The system should run as a web application.
- The system should be able to support Java.
- The system should be able to keep track of system log.

1.5 Expected Benefits

This project provides the following expected benefits:

- Providing a tool to identify code clone in software systems.
- Helping software developers to be aware of redundancy source code in their software system.
- Performing code clone detection easier and more convenient.
- Decreasing false clones in clone detection result.

1.6 Organization of the document

The document consists of 6 parts that are Introduction (Chapter 1), Background (Chapter 2), Analysis and Design (Chapter 3), Implementation (Chapter 4), Evaluation Results (Chapter 5), and Conclusion (Chapter 6). The Introduction chapter includes motivation, problem statement, the objective of the project, scope of the project, expected benefits, and organization of the document. The Background chapter describes the overview of the project, which has fundamental and related work. Analysis and design chapter contains work procedures, which are methodology, system architecture, structure chart, and system analysis. Implementation includes building an ML code clone detection engine, using trained ML models for clone detection, and building a web application. Evaluation Results consists of the evaluation of Merry clone detection engine using BigCloneBench, evaluation of Merry clone detection engine on a real software project, and evaluation of Merry web application by users. The last chapter is conclusion that includes conclusion, problem and limitations, and future work.

CHAPTER 2
BACKGROUND

2.1 Fundamentals

The fundamental section is to provide the basic knowledge of the project including code clones, an overview of clone detection techniques, machine learning, and web application.

2.1.1 Code Clones

Roy et al. [4] have defined the word code clone as a fragment of code that appear several times in software during software development. Code clones are not only the copy-and-paste segments of code but they also contain minor modifications of code. Another definition of code clones was given by Bellon et al. [5], Clone pair is two similar code fragments based on the definition of similarity. The research shows that about 7% to 23% of all over software system consists of clone fragments. However, the cloning code in a software project can be harmful for several reasons. For example, if a code fragment has bugs and clone to other places of software, Debugging will consume developer time. Copy of code fragments can increase workload when enhancing or adapting. Detecting code leading to multiple benefits between developing software such as detect plagiarism, quality analysis, virus detection, etc. In the study of clone detection, Roy et al [4]. give a definition to use in this study as follows:

1. Clone Fragment or CF is a sequence of code line that includes with or without comments.

2. Code clone is two code fragment or CF1, CF2. If code fragment number one is a clone of code fragment number two, we can call them a clone pair and can represent in (CF1, CF2). Furthermore, if many code fragments are similar, we can call it clone class or clone group.

Clone Types

Clones can be categorized into 4 types according to their syntactic and semantic similarity. For type 1 to type 3 is the syntactic-based and type 4 is functionality-based.

- Type 1: Code fragments that are identical except for layout, white space, and comments.
- Type 2: Code fragments that are identical except for literals, identifiers, data types, layout, white space, and comments.
- Types 3: Similar clone fragments that contain added, changed, and removed some statement.
- Type 4: Code fragments that have the same computation but different in syntax or algorithm.

Code Clone Detection Process

There are 3 fundamental steps of all existing clone detection tools and techniques in performing clone detection process.

1. Pre-processing

This is a preparation step for raw codes to detect clone easily. The preparation contains three phases. Firstly, removing uninteresting parts use to filter raw codes into a single language that can detect a clone. For example, some of JAVA code has SQL embedded. Determining source units is the next phases by separating into disjoint fragments. The last one is to determine comparison units/granularity for massive source units, so we need to separate it more to make it even smaller.

2. Transformation

Transformation earn the fragment of code that call units of comparison. Then, transforming the unit of comparison to appropriate inter-mediation for comparison. This method to convert in reverse engineering community often called extraction. However, some code clone detection tools support the process called normalization, which is the process that will make source code simple normalize form such as remove white space and comment, involving source code transformation. By include normalization step, it could be done before or after extraction.

2.1 Extraction

Extraction is the method that transforms source code to be suitable for comparison algorithm. Extraction has multiple methods depend on tools.

- Tokenization: Tokenization is the process that changes the line of source code into token form based on lexical rule of programming language. It also removes whitespace between these processes.
- Parsing: Parsing is the process that transforms a line of source code into a tree which is call parse tree or abstract syntax tree (AST). Then, comparison algorithms with look for similar subtrees to mark as clones.
- Control and data flow analysis: This is the process that will generate program dependence graphs or PDGs from source code. In PDGs, it has a node that represents the statements and conditions of the program and edge represent controls and data dependencies. Subgraphs show clone when getting through the comparison algorithms.

2.2 Normalization

Normalization is an optional step to make source code in a suitable form for detecting clones.

- Removal of white space: This step is to remove white space from code.
- Removal of comments: This step is to remove or ignore all comments.
- Normalizing identifiers: This step is to replace difference identifier in to single identifier. It helps to detect Type2 clones.
- Pretty-printing of source code: This step is to make the source code into a standard from which is removes difference in spacing and layout.

- Structural transformation: This step makes the source code minor modify but still has the same syntax such as remove static in the C programming language.

2.3 Match detection

When receiving the transformed code, it will go through the comparison algorithm to find matches. Popular matching algorithms for clone detection are suffix-trees, dynamic pattern matching (DPM), and Hash-value comparison.

2.4 Formatting

After the match detection is done, the result of matching algorithm will compare with the original source file.

2.5 Post-processing / Filtering

This is a process to rank or filter clones which has two popular methods.

- Manual analysis: This is a method that filter clone by human expert. They will focus on false positive clones.

- Automated heuristics: This is a method that will automate rank or filter from characteristic of clones.

2.6 Aggregations

This process will group clone group into clone class to reduce resources.

Overview of Clone Detection Techniques and Tools

Based on analysis applied on the source code, techniques can be classified into four main categories which is textual, lexical, syntactic, and semantic.

1. **Textual approaches** or text-based techniques are clone detection techniques that require little or no transformation/normalization of code before directly comparing them together. Examples of a technique for text-based detection include fingerprints, scatter plots or dot pot, etc. NICAD [6, 7], Simian [8] and Marcus [9] are the example of text-based techniques.

2. **Lexical approaches** or token-based techniques is the technique that transforms the source code into tokens using compiler-style lexical analysis. Then, scanning tokens for group same token to be clones. This technique mostly found type1, type2, and type3 in clone detection. This is the example of token-based techniques including CP-Miner [10], iClones [11], and CCFinder [12, 13].

3. **Syntactic approaches** focuses on parse tree or abstract syntax trees (ASTs) which the process can use tree matching method or structural metrics to find clone.

   - Tree matching approaches or tree-based techniques: It using tree characteristic to find clones such characteristic as subtrees, Variable names, literal values and other leaves (tokens).

   - Metric-based approaches: This approaches use metrics applied to code fragments and vectors of metrics rather than code or ASTs directly.

   The example of tools that use this techniques are CloneDr [14], SimScan [15], and Deckard [16].

4. **Semantic approaches** use a program dependency graph (PDG) to visualize the control flow and data dependencies of the source code in the project. By using this static programming analysis makes semantic approaches more accurate than simple syntactic similarity. Davey [17] is the tool that using semantic approaches.

5. **Hybrid approaches** This is a combination between syntactic and semantic approaches. Duplix [18], Gabel [19], and Komondoor [20] are the example of hybrid tools.

2.1.2 Machine Learning

Machine Learning or ML is a tool to improve the progress of data analysis. It has self-learning ability without entering the commands of the programmer. ML demonstrates that a computer can only learn from data to produce accurate results. They are generally divided into 2 categories, which are supervised and unsupervised.

Supervised learning uses a data set that the labels are already known. It is categorized into (1) regression, which predicts continuous results and (2) classification problems that predict discrete results. Decision tree, random forest, and support vector machine are the example of a supervised model.

On the other side, **unsupervised learning** requires only a data set without any labels including (1) clustering, which distributes data into smaller groups by their similarity and divergence and (2) association which is to discover the rules and relationship of the data. An example of unsupervised learning is k-means. In our project, we will use the supervised Machine Learning model because we want to classify clones into true clones and false clones.

2.1.3 Web Application

A web application is one kind of online service. It is a system that provides a user interface through a web browser [21]. Typically, the architecture of web application is a client-server system including some remarkable distinctions. Deployment is a significant advantage of a web application. Most deployments for a web application is about setting up the components on the network on a server-side. On the client-side, no additional software or configuration is needed [22].

Using the web application, users do not need to set an environment to execute a code clone detection tool. On the other hand, traditional clone detection tools are complicated to use because they have to run by a command in the command line, so the user has to know the command and tool's parameters to execute the tool. Moreover, the result that difficult to understand in traditional command-line based tools can be visualized into graphic in a web application to understand the result more comfortable. Therefore, improving user performance is one of the reasons that web application involves our code clone detection project.

2.2 Tools and Techniques

This section explains several tools and techniques which are applied into the project.

2.2.1 GitHub

Git is version control that keeps file changes in the project. GitHub is the largest code archives in the form of Git website and developers community. There are more than one hundred million repositories and forty million developers[1] in GitHub [23] Around the world, GitHub are often used to store open source projects, for example, Bootstrap, Node.js, Angular etc. Several projects in GitHub can be able to see and also can be shared with other people. It is therefore very effective if working as a team.

2.2.2 GitHub API

GitHub provides API (Application Programming Interface – an interface that allows third-party programs to request for data from a host or a data holder) for third-party companies or coding projects. API gives developers to connect with GitHub system. API also provides access to some features. For example, GitHub API gives privilege to third-party system which authorized users to manage repository such as create, view, deleted their repository through a third-party system. Another example is authorized with GitHub. This is giving third-parties system about necessary information that store in GitHub ID.

2.2.3 Node.js

Node.js is the name of an open-source framework for the creation of a web-server using JavaScript language. Traditional JavaScript is a language for client-side web development. Therefore, it needs to cooperate with another language, like PHP, to work on the server-side. Node.js is the open-source software built by Google. It can work on the server-side more suitable than other server-side software because it is written combine with C language, C++, and JavaScript. For the benefit of writing in C language and C++, This makes Node.js super fast and supports multiple operating systems. Moreover, including JavaScript makes Node.js accept the JSON file that is a common data interchange of the website. Node.js has a feature that is non-I/O blocking. For example, compare with PHP, The non-I/O blocking makes Node.js execute a command in parallel and has a callback function to report back which success or failure. In contrast, PHP

[1] Data as of January 2020

executes a command when the previous command finishes only.

2.2.4 MongoDB

MongoDB is an open-source document database [24]. There is no relation (i.e., NoSQL), and it stores data as JSON (JavaScript Object Notation) objects. The data is stored in terms of key and value, which is called a document. A collection of many documents in MongoDB is stored as a collection. Moreover, the collection is schema-less, which means there is no need to define any structure. Since MongoDB is flexible and uncomplicated, several well-known companies are using MongoDB, such as Baidu and Adobe Experience Manager.

2.2.5 WEKA

Weka is the Java open-source machine learning application that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It provides several features which use in Machine Learning techniques, for instance, data manipulation, model training, and model evaluation. It was created by the Machine Learning Group of the University of Waikato, New Zealand [25]. In this project, we use Weka for training the machine learning model.

2.2.6 Java Parser

Java Parser is the tool that decomposes the java source code into smaller elements such as classes, methods, or parameters. These smaller elements are extracted from an Abstract Syntax Tree (AST) of given source code after analyzed by the parser [26]. We aim to detect the clone in method level which is a suitable granularity in clone detection because each method performs only one function. However, if we compare at the file level that performs several functions in a file which is too rough to detect clones. This level of clone detection is also used by several existing clone detection tools, for instance, CCLearner, Oreo, and NICAD. Moreover, the ground truth data from BCB are method level.

We use the Java Parser for extracting methods from Java source code in both data preparation and code clone detection steps.

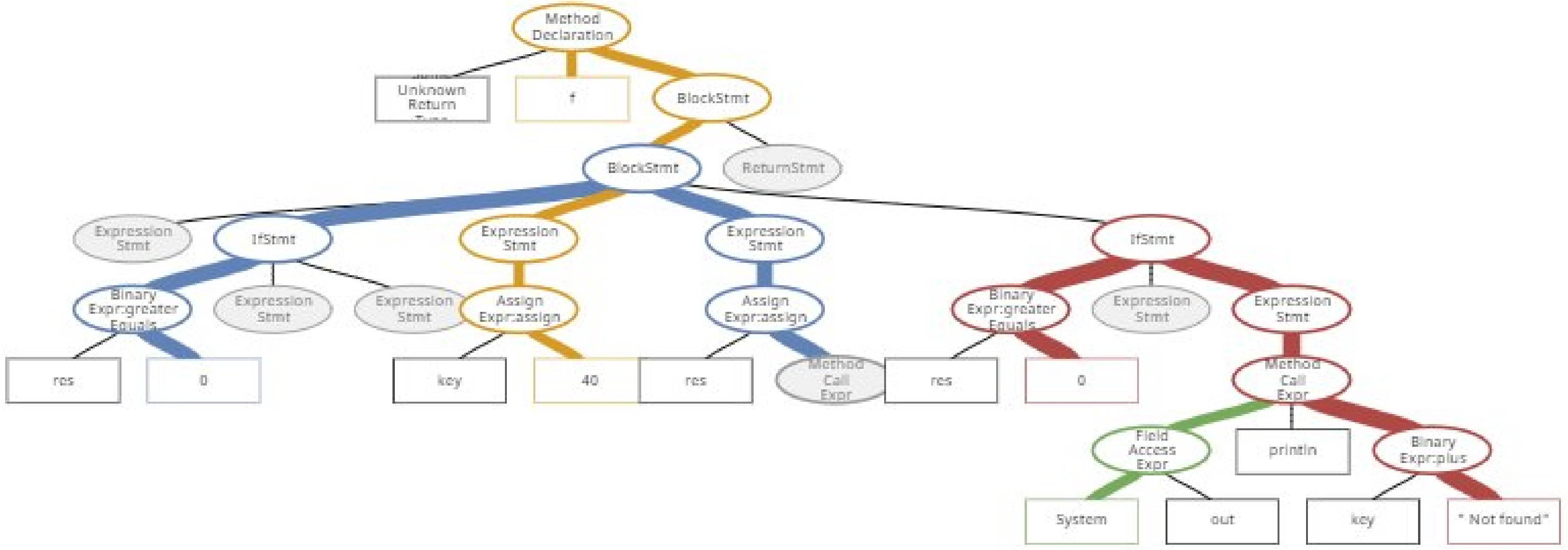
2.2.7 Java Tokenizer

Java Tokenizer tool called “ANTLR” is used in our project. ANTLR is a parser generator for reading, writing, executing, or converting structured text or binary files. After we get methods from JavaParser, we use ANTLR tokenize Java methods into tokens or words with is needed for creating metrics that are used in the model training and clone detection steps.

2.2.8 code2vec

code2vec [27] is a neural network model that creates a fixed-length vector that represents the semantic of a given Java source code snippet. It can also give the prediction of the name according to the Java method body such as `sort`, `find`, or `count`. The main idea of code2vec predicting source snippet vectors is representing source code as a set of AST paths before aggregate all paths in the set by using neural attention. Because code2vec can represent the semantic of the source code, we use code2vec to enhance our machine learning performance by using the vector from code2vec as our semantic features.

Figure 2.1: Set of AST path of finding method



Listing 2.1: Java binary search source code

```
{
   int res = Arrays.binarySearch(arr, key);
   if (res >= 0)
      System.out.println(key + " found at index = " + res);
   else
      System.out.println(key + " Not found");
   key = 40;
   res = Arrays.binarySearch(arr, key);
   if (res >= 0)
      System.out.println(key + " found at index = " + res);
   else
      System.out.println(key + " Not found");
}
```

Figure 2.1 represent the AST paths of the binary search source code in Listing 2.1 that visualize is retrieved from code2vec.org (the official code2vec website) [28]. Each color on the tree represent the path of data flow of the source code that will be converted into a vector. Then, code2vec uses all path vectors to predict the final vector of source code by code2vec's algorithm.

2.3 Literature Review

The literature review section is a summary of the researches which are related to the project.

2.3.1 Syntactic Code Clone Detection Technique

Gode et al. [11] present an incremental clone detection technique that detect clones in several revisions of the system to analyze the evolution of clones. This incremental clone detection technique based on suffix trees and use for token-based clone detector. The token-based detector performs the lexical analysis to the source code to convert source code into the sequence of tokens. A token has type, which is the lexical category of the token, and value that is the textual presence of the token. Considering the token stream, the challenge of detecting exact clones relies on detecting equivalent token sequences. Types of clones that can detect using this technique are Type-1 clone (exact clone), Type-2 clone (parameter-substituted clone), and Type-3 (modified clone). Even though only Type-1 clones and Type-2 clones can be detected by using suffix trees, Type-3 can be detected by combining Type-1 clones and Type-2 clones in to the larger clones with a certain threshold of lexical gap.

Figure 2.1 shows the structure of the suffix tree that is used in this technique, which is generalized suffix tree (GST) that represents the suffixes of set of code strings. GST also have property that suffix links of leaves always indicate to a leaf that represents a suffix of the same string. A string is inserted into a set of strings by adding its suffixes from longest to shortest to the GST. Due to the fact that files change from one revision to the next revision, the tokens of these files have to be added or removed from the token table and the set of strings. Their incremental clone detection algorithm reuses the data resulting from the previous revision analysis which contain token table, GST, and the set of clones. The new clones that come from file changing also retrieved from the updated GST. From the experimental they compare the incremental (iClones) and non-incremental (traditional token-based clone detection) technique in two scenarios; the first scenario is the clone detectors are part of an integrated development environment (IDE), and another scenario is evolution, show that iClones require only 25% run time of traditional token-based clone detection but it also require 50% memory space than traditional token-based clone detection for both scenarios. To conclude, they implement the incremental clone detection technique based on generalized suffix tree call "iClones" that use for token-based detection tool which can improve the run time when compared to traditional token-based clone detection technique.

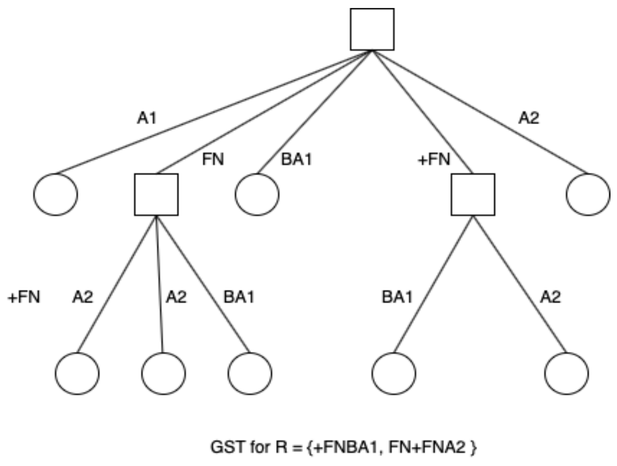


Figure 2.2: Generalized Suffix Tree (GST)

SonarQupe [29] is the platform that developed and service by SonarSource. This platform is open-source which features are focus on inspection of code and automation reviews. This platform also provide detect bugs, code smell which is part of codes that could be a deep problem, security vulnerability. SonarQupe also support more over 20 languages such as Java, C/C++, C#, etc. However, one of SonarQupe features relates to code clone detection called duplicated code. This feature focus on detection duplication in codes and it can ignore some variables and indentations.

2.3.2 Using Machine Learning To Improve Existing Code Clone Detection Technique

Koschke et al. [30] presented the clones rate analysis of 7,800 open-source C or C++ projects. They used their own token-based clone detector call “cpf” to detect clone. This detector detect clones by creating suffix tree and detect all the duplicating subsequent of token in the suffix array. In their researching period their clone detector can detect only type-1 and type-2 clones. So, type-3 clone in their research are from combining type-1 and type-2 clones into one segment. After getting these clone candidates the detector create code metrics which are number of tokens, number of parameters, clone type, the number of distinct token types, fraction of non-repetitive tokens, parameter overlap, parameter consistency, degree of valid references, and fraction of repeated parameter to be the feature of decision tree filter that automatic calibrate by machine to improve precision without losing recall. In the result they can determined only 20 percent of the open-source projects to have no type-2 clone and 44 percent of projects contain at least 1 type-1 clone of at minimum of 1000 tokens.

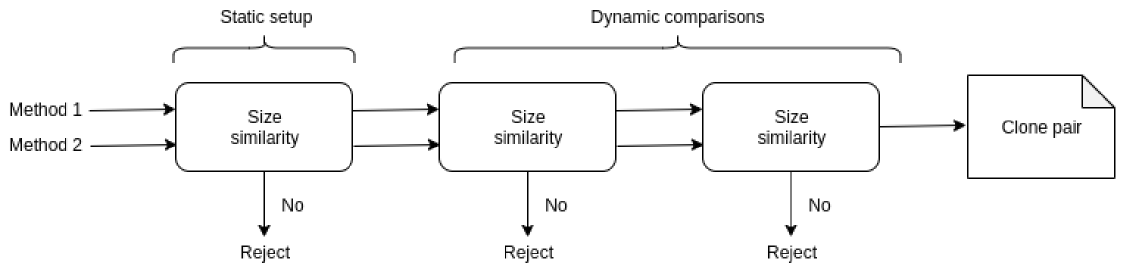


Figure 2.3: Overview Diagram of Oreo's Clones Detection Pipeline

Saini et al. [31] invented a code clone detection tool name “Oreo”. Oreo was invented to improve a detecting clone in Twilight zone and able to process large dataset. Twilight zone is define to be a meaning of challenging Type3 clone which contain moderately type3 and onward. As the figure 2.1 shows, Oreo use size to be first heristic to measure clone between two methods. This is specific for clone in twilight zone. The Action Filter is a a next process after using size. This is due to low lexical and syntactic similarity in Twilight zone. The action filter use "Action token" to capture semantics of methods. For example, name[1] action token will capture it as ArrayAccess. On the other hand, if name[i+1] action token will difference capture it to ArrayAccessBinary. Not only Action token that added in this process but it also store with "freq". "freq" is the number of that action token appear in the method. In this paper, it define in set of < t , freq > which t is action token. After, track action token it will filter by metric hash. If metric hash matched, it is classified into type1 and type2. However, if it not match with metric hash, it will pass though neural networks or deep learning methods to detect type3 and type4 clones. Deep learning has characteristic to handle the vector so Oreo uses Siamese architecture model to train. The input model is a 48 dimensional vector that come from 24 metrics. In Siamese, two inputs are sent into identical subnetworks of Siamese. Then, all of output from subnetworks are send to the comparator for concatenated. Classification unit which is output of comparator will multiple with input to be final output in range between 0 and 1. In this paper, they classify codes that have value above 0.5 to be clones. They involve Siamese can prevent overfiting and it can support the large number of training set.

Arammongkolvichai et al. [32] present an approach for increasing the precision of code clone detection using machine learning techniques. This approach is working like a filter to filter the false-positive results out out from clone result. The filter are created from decision tree that training from 19 clone metrics. In their experiment, they apply this filter into clone detection tool called "iClones" and compare the result between result of iClones and iClones with filter. They found that using filter on iClones can increase 4% of precision from 94% to 98% by filtering the false-positive result out.

2.3.3 Using Machine Learning For Code Clone Detection

Li et al. [33] presented the first solely token-based clone detection approach using deep learning which called CCLearner. CCLearner applies the concept of deep learning on known true clones and false clones for training. CCLearner uses the training and testing data from BigCloneBench [34] that contains ten folders by divided folder 4 for training and the other folders for testing. The amount of data is 22,663 true clone and 22,663 false clone pairs for training and 1,581,218 clone pairs for testing. In the training phase, they trained the model with true and false clone pairs. Then extracts the similarity vectors and give them to a deep neural network (DNN) which include two hidden layers and run 300 iterations based on their experiment to train a binary classifier. For the testing phase, firstly they extract methods from source code by using Eclipse ASTParser after that compare method pairs and tagged pairs as clones or non-clones. An evaluation session, CCLearner compare the efficiency with three popular clone detection tools that are SourcererCC [35], Nicad [36], and Deckard [16]. The result is that CCLearner has higher recall than SourcererCC and Nica and higher precision than Nicad and Deckard. To sum up, CCLearner performs highly effective in detecting clones using deep learning which is evident from high precision and recall.

White et al. [37] represent a new way to detect clone that is learning-based clone detection techniques using deep learning AST-based and Greedy. All source code is collected as terms or fragments and mined from the repository. The first process is deep learning code in lexical level that transform the original code to be a suitable similar forms or terms. Each term in a fragment are mapped to similar vectors. The second process is deep learning code in syntactic level which design the learning-based techniques to learn features at different levels of granularity. Lastly, They use neural networks paring with lexical analysis and recursive neural networks paring with syntactic analysis. The data that they used consist of eight real world Java system which are ANTLR 4, Apache Ant 1.9.6, ArgoUML 0.34, CAROL 2.0.5, dnsjava 2.0.0, Hibernate 2, JDK 1.4.2, and JHotDraw 6. In precision results, Greedy perform better performance than AST-based with almost 100% of both file-level and method-level in eight Java system.

2.3.4 Dataset for Evaluating Code Clone Detection Techniques

Svajlenko et al. [34] presented the benchmark, called **BigCloneBench**, of known true and false clones from IJaDataset, which is a big data inter-project Java repository and demonstrated how the benchmark could be used to measure the recall and precision of clone detection techniques. BigCloneBench built by mining for code snippets. It is a segment of code stored as (l, s, e) format by l represents the source file, s represents the start line, and e represents the end line. The methodology is that it first selects the target functionality; for example, a binary search. Then creates search heuristic to identify possible implementation followed by forming a specification, sample snippet, and search heuristic. After that, the search heuristic is performed for every snippet to find the candidate snippets. The last step is to tag all candidate snippets, which snippet is true or false positives by manual. A set of candidate snippets which can implement the target functionality were tagged as true positives, and candidate snippets that do not implement the target functionality were tagged as a false positive. True clone pairs calculate from (s+p)(s+p1)/2 by s stands for sample snippets of particular target functionality and p stands for true positive snippets. False clone pairs calculate by sample snippets of target functionality multiply with false positives snippets. The result from 10 functionalities with 60,000 snippets can detect approximate 6,000,000 true clone pairs and 260,000 false clone pairs. Besides, the benchmark can also measure recall and precision.

2.4 Comparison to Related Work

As discussed in the Literature Review, there are several existing clone detection tools including Software Clone Rate, iClones, iClones + Filter, Oreo, and Sonar. Machine Learning Filter (ML Filter), Machine Learning Engine (ML engine), Web Application and Clone Type Detecting are the function of code clone detection tools. As report by the existing tool comparison table (Table 2.1), each tool is capable of performing different functions.

The fist one from Koschke et al.[30] is able to detect clone type 1,2, and 3 using machine learning. iClones [11] does not use the machine learning but it also detect clone type 1,2, and 3. iClones + Filter [32] is similar to iClones but it apply the machine learning. Oreo[31] is only one existing tool that can detect type 4 clone using machine learning. The last existing tool is Sonar which has web to show the visualization of clone result. Our clone detection tool, Merry, is able to detect Type 1,2, and 3 using machine learning. Moreover, Merry tool provide the web application for user.

Table 2.1: Comparison with Existing Tools

Tool/Technique	SyntacticFeature	SemanticFeature	ML Filter	ML Engine	Web UI	Type 1	Type 2	Type 3	Type 4
iClones (Gode et al.) [11]	●	-	-	-	-	●	●	-	-
Koschke et al. [30]	●	-	●	-	-	●	●	◒	-
iClones+Filter (Arammongkolvichai et al.) [32]	●	-	●	-	-	●	●	●	-
Oreo (Saini et al.) [31]	●	-	●	-	-	●	●	●	◔
Sonar [29]	●	-	-	-	●	●	●	-	-
CCLearner (Li et al.) [33]	●	-	-	●	-	●	●	●	◔
Merry	●	●	-	●	●	●	●	●	◔

(●=fully function, ◒=function with limitations)

CHAPTER 3
ANALYSIS AND DESIGN

Analysis and Design chapter illustrates the system design of the project from the overview to the detailed steps in each process.

3.1 System Architecture Overview

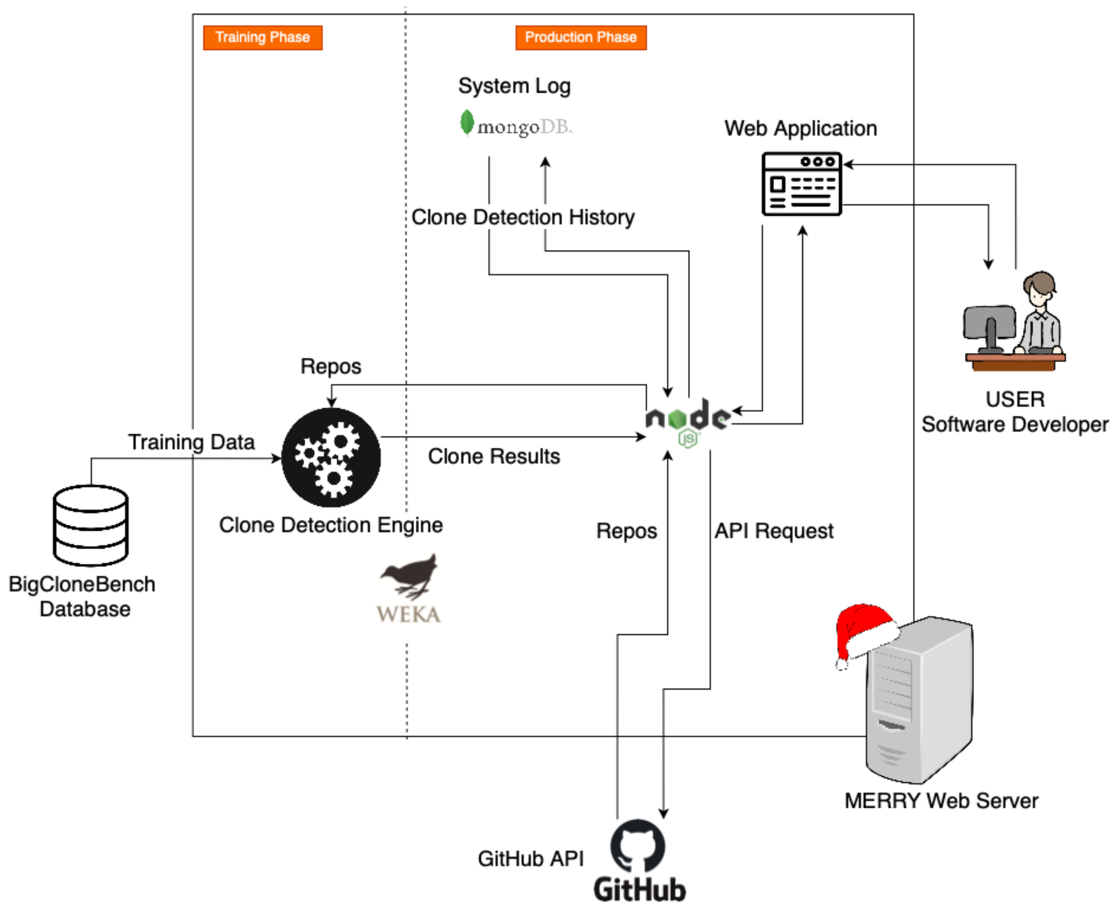


Figure 3.1: System Architecture

System Architecture (Figure 3.1) shows the overall of our web-based code clone detection tools using machine learning, which is called Merry. There are various functions inside the system which consists of training phase and production phase.

In the “training phase”, the starting process is an analysis of the clone database as training data. The database which was used in the project is BigCloneBench database. It is the largest and the most reliable clone database. After all training data are prepared, the training data was sent to the clone detection engine for training the model. That is when the training phase finishes.

The second phase is the “production phase”. First, users have to log in to the system with their GitHub account to analyze their repository that they want to detect the clone. In the meanwhile, the system also collect the clone detection history into the database.

3.2 Use Case Analysis

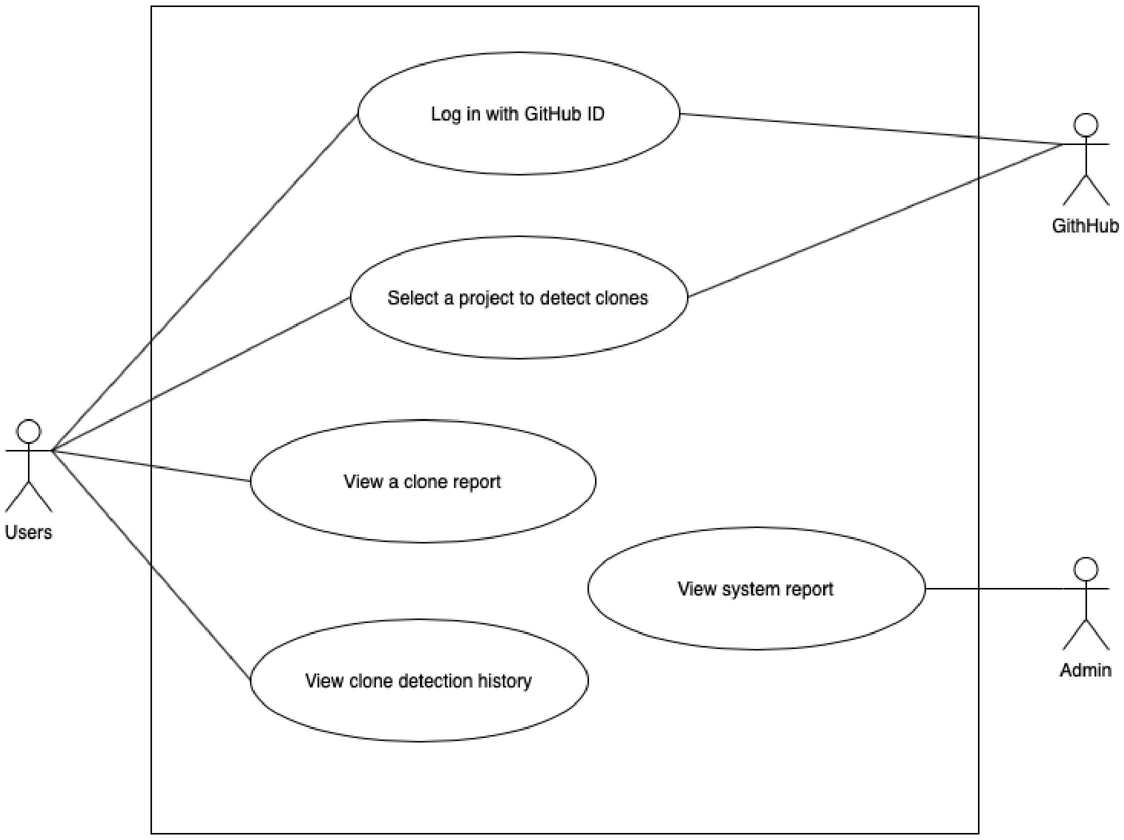


Figure 3.2: Use case Diagram

The use case diagram (Figure 3.2) represents about users interact with the project. In this project, our users are separated into two kinds. First is normal users that want to detection clone in their project. Our system will provide such features as login with GitHub, selecting a project to detect clones, viewing clone detection history, and viewing system log(for themselves). Most of data like user’s repositories come from GitHub after users give permission.A clone report and system log come from our system. Another kind of a user is admin. admin can also view system log. However, system log of admin will be overall of users.

3.3 Structure Chart

Structure Chart (Figure 3.3) represents all modules in the Merry system. There are 4 main modules consists of data preparation, model training, clone detection, and report generation. Moreover, the first 3 main modules also have the sub modules. Data Preparation includes 4 sub modules which are parsing the source code, creating the method dictionary, selection clone pairs, and creating training data. Model training made up of 3 sub processes which are training data tokenization, extracting metrics, and training model. The last module is clone detection which consists of getting the source code in repos files, parsing method, mapping method pairwise, tokenization, extracting clone metrics, and clones classification.

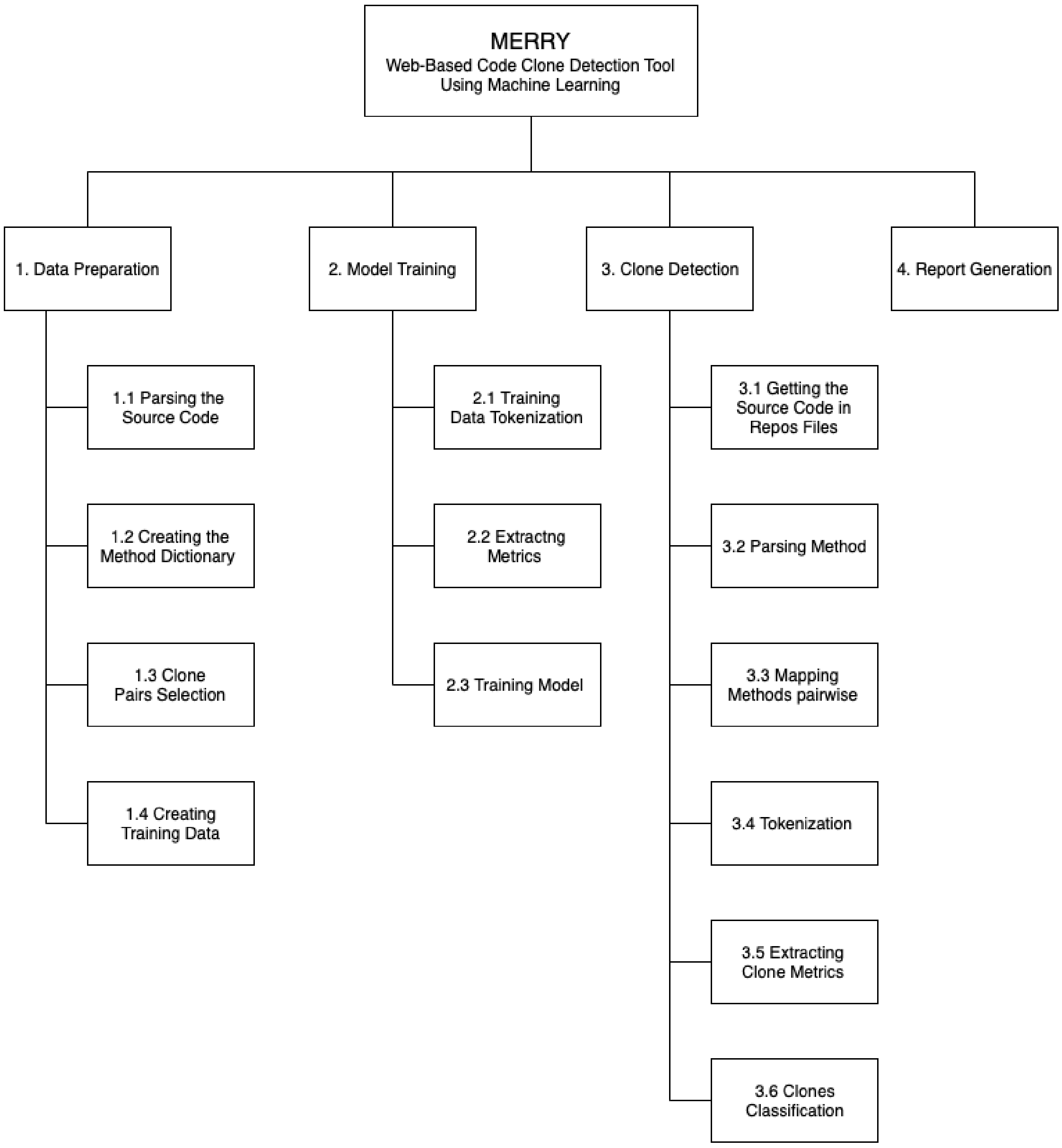

MERRY
Web-Based Code Clone Detection Tool Using Machine Learning
1. Data Preparation
2. Model Training
3. Clone Detection
4. Report Generation
1.1 Parsing the Source Code
1.2 Creating the Method Dictionary
1.3 Clone Pairs Selection
1.4 Creating Training Data
2.1 Training Data Tokenization
2.2 Extractng Metrics
2.3 Training Model
3.1 Getting the Source Code in Repos Files
3.2 Parsing Method
3.3 Mapping Methods pairwise
3.4 Tokenization
3.5 Extracting Clone Metrics
3.6 Clones Classification

Figure 3.3: Structure Chart

3.4 System Analysis

System Analysis of the project was represented by data flow diagram level 0, level 1, and level 2.

3.4.1 Context Diagram

According to context diagram(Figure 3.4), there are 3 external entities that get involved to the Merry system. BCB or BigCloneBench database send clones ground trust and source code files to the system. User send the clone detection request to the user then the system automatically send API request to GitHub. After that GitHub send the repos to the Merry system. Finally, User will receive the clones report and visualization.

3.4.2 Data Flow Diagram Level 1

Data Flow Diagram Level 1(Figure 3.5) show the all the flow inside our system starting from receive source code files and clone Ground-truth data to prepare the training data for train the machine learning model that use in clone detection process. After the model is trained the system is ready to detect clones. When get the clones detection request from user, system will sent API request to GitHub for repository that user want to analyze clones. Then the system will generate visualization and report for the clones result.

3.4.3 Data Flow Diagram Level 2

Prepare Data Process

Preapare data process(Figure 3.6) start with parsing the source code from files BCB database into list of methods to create the method dictionary for easily to get the ground-truth clone pairs and non-clone pairs for generate training data.

Train Model Process

The model training process(Figure 3.7) is starting from the training data tokenization into list of token for extract code syntactic metrics to use as the features of the model. After that train the model with all features that already gather.

Detect Clones Process

Detect Clones Process(Figure 3.8) start from get all Java source files in the repository then parse all files to get all methods by JavaParser before pairwise all methods. Then tokenize each pair in the list to extract the code syntactic metrics to classify clones.

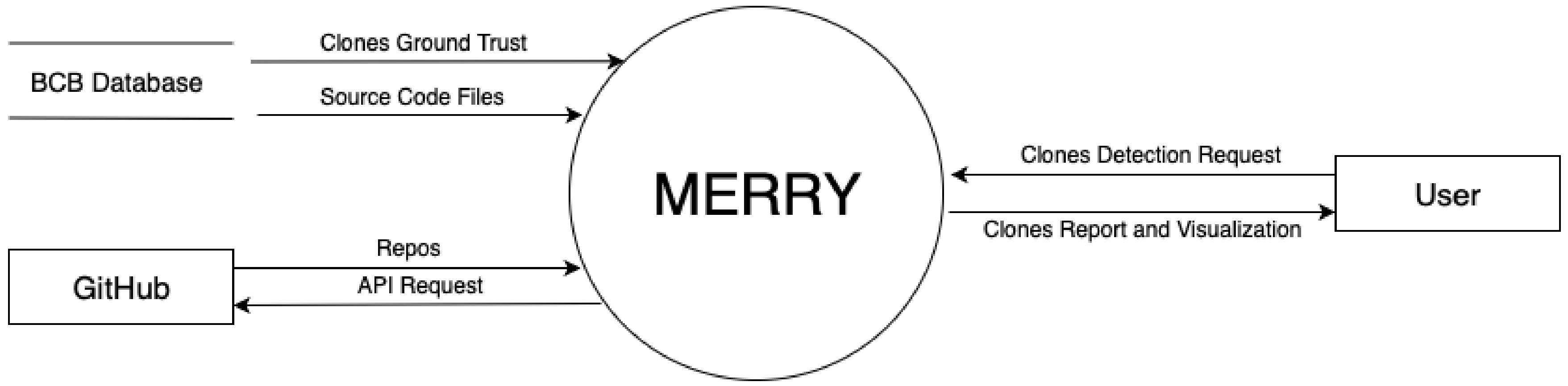


Figure 3.4: Context Diagram

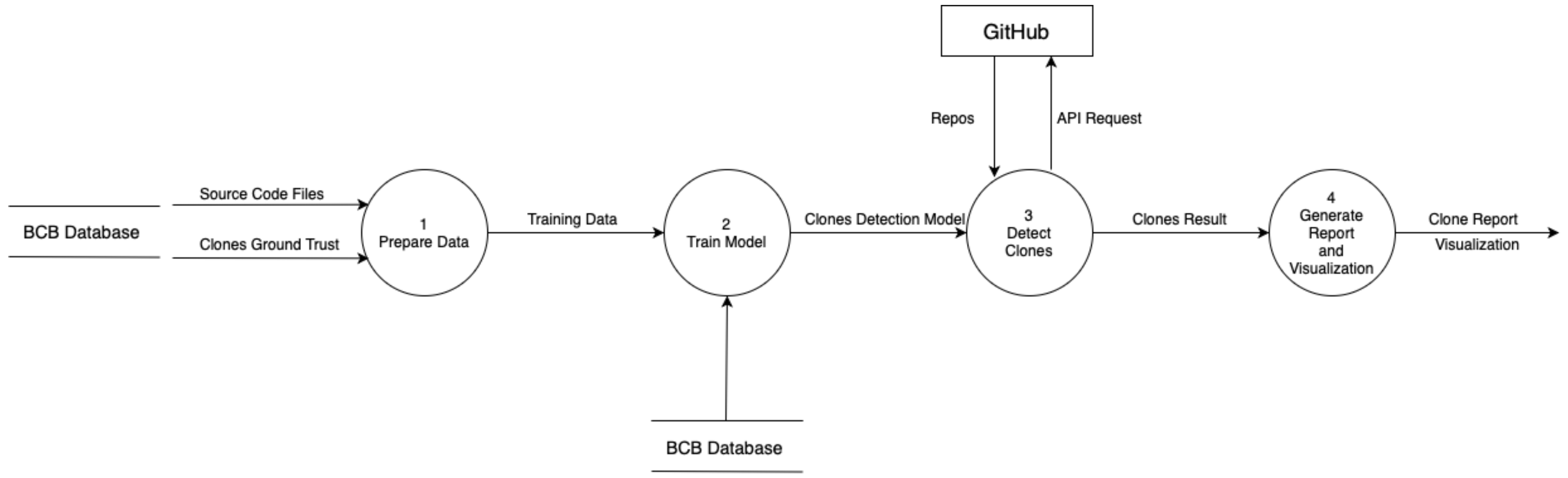


Figure 3.5: Data Flow Diagram Level 1

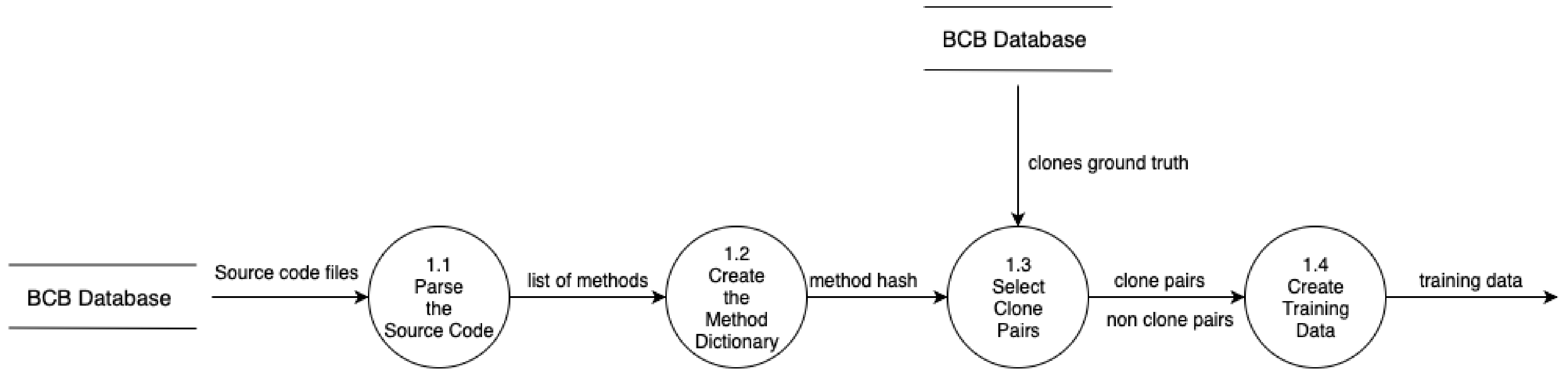


Figure 3.6: Prepare Data Process

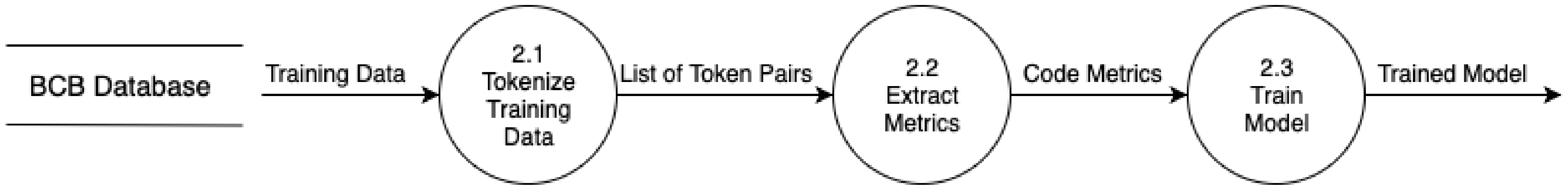


Figure 3.7: Train Model Process

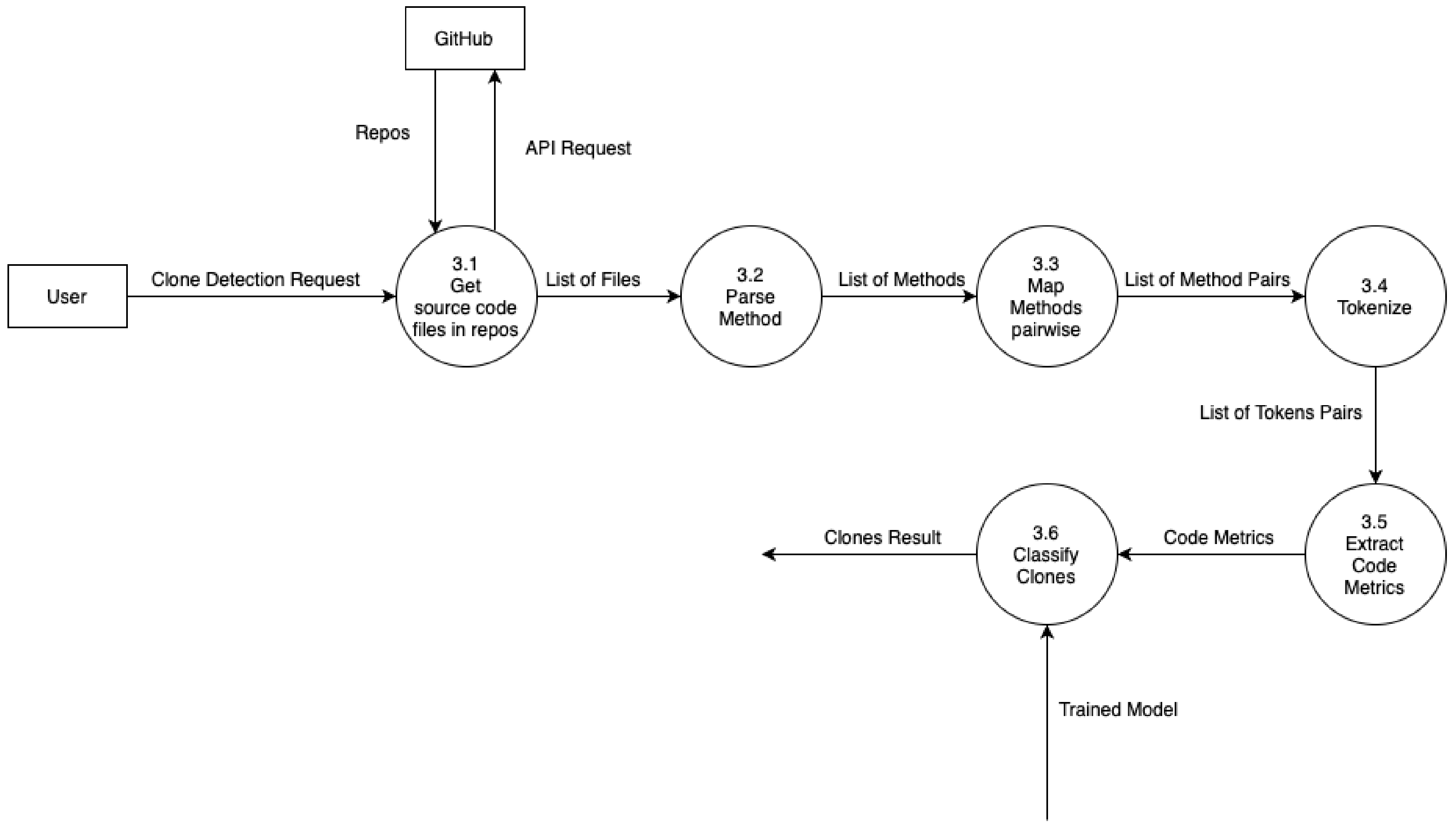


Figure 3.8: Detect Clones Process

CHAPTER 4
IMPLEMENTATION

In this section, there are 3 parts divided by project components which include clone detection engine part, web application part, and data collection part.

4.1 Building ML Code Clone Detection Engine

This step (as shown in Figure 4.1) is the step to train machine learning models. It includes data collection and preparation, code metrics extraction, ML model selection, and training.

4.1.1 Data Collection and Preparation

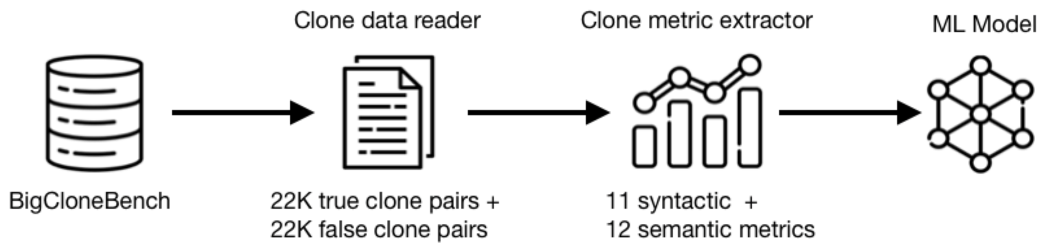


Figure 4.1: Training process

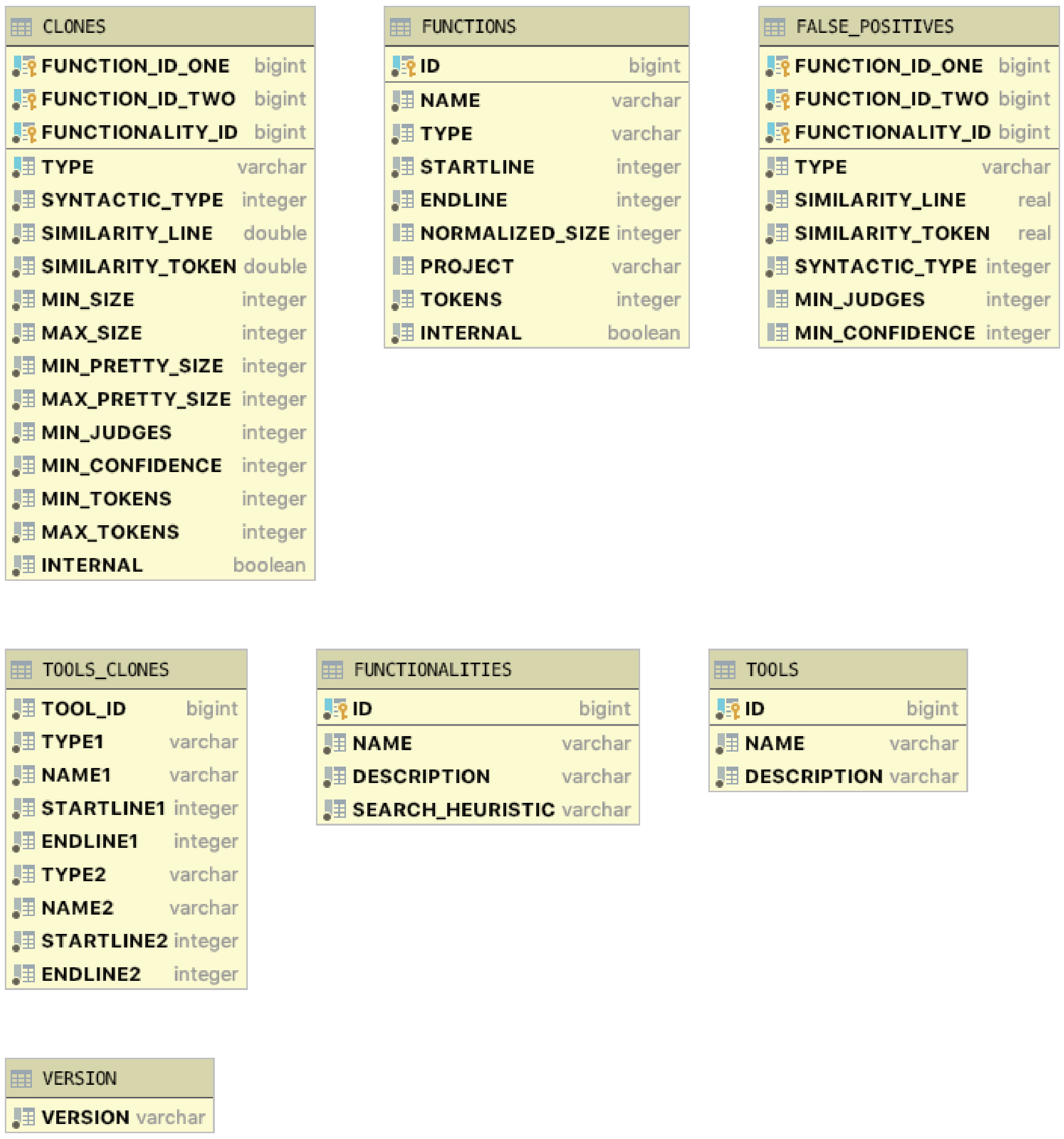
Data set

We used BigCloneBench [34] (BCB) which is clone detection benchmark. The benchmark is a large data set of clones that tagged as true and false clone pairs and popular in the clone detection approach. It is divided into 2 parts, which are Java source code files and code clone database.

Java Source Code Files: As shown in Table 4.1, BCB have 40,528 Java source files. There are 10 folders separated by their functionality starting from folder#2 to folder#11.

Code Clone Database: BCB contains over six million clone pairs with 10 functionalities stored in the PostgreSQL database. Thus, we wrote SQL statements to get the clone pairs dataset. The BigCloneBench database (Figure 4.2), which will be used as training data, has 7 tables. The 3 mains tables are `clones`, `functions`, and `false_positives`. After evaluation and analysis of the BigCloneBench structure, we found that the clone has been given the similarity score in range [1,0]. Basically, clones have 4 types [4], as discussed in Section 2. However, BCB [33] have separated clones into even more fine-grained levels of 6 categories as follows:

- Type-1 (T1) has similarity score equal to 1 and syntactic type is 1
- Type-2 (T2) has similarity score equal to 1 and syntactic type is 2
- Very strong type-3 (VST3) has similarity in range [0.9,1)
- Strongly type-3 (ST3) has similarity in range [0.7,0.9)
- Moderately type-3 (MT3) has similarity in range [0.5,0.7)
- Weakly type-3 or type-4 (WT3+4) has similarity in range [0,0.5)



CLONES	
FUNCTION_ID_ONE	bigint
FUNCTION_ID_TWO	bigint
FUNCTIONALITY_ID	bigint
TYPE	varchar
SYNTACTIC_TYPE	integer
SIMILARITY_LINE	double
SIMILARITY_TOKEN	double
MIN_SIZE	integer
MAX_SIZE	integer
MIN_PRETTY_SIZE	integer
MAX_PRETTY_SIZE	integer
MIN_JUDGES	integer
MIN_CONFIDENCE	integer
MIN_TOKENS	integer
MAX_TOKENS	integer
INTERNAL	boolean

FUNCTIONS	
ID	bigint
NAME	varchar
TYPE	varchar
STARTLINE	integer
ENDLINE	integer
NORMALIZED_SIZE	integer
PROJECT	varchar
TOKENS	integer
INTERNAL	boolean

FALSE_POSITIVES	
FUNCTION_ID_ONE	bigint
FUNCTION_ID_TWO	bigint
FUNCTIONALITY_ID	bigint
TYPE	varchar
SIMILARITY_LINE	real
SIMILARITY_TOKEN	real
SYNTACTIC_TYPE	integer
MIN_JUDGES	integer
MIN_CONFIDENCE	integer

TOOLS_CLONES	
TOOL_ID	bigint
TYPE1	varchar
NAME1	varchar
STARTLINE1	integer
ENDLINE1	integer
TYPE2	varchar
NAME2	varchar
STARTLINE2	integer
ENDLINE2	integer

FUNCTIONALITIES	
ID	bigint
NAME	varchar
DESCRIPTION	varchar
SEARCH_HEURISTIC	varchar

TOOLS	
ID	bigint
NAME	varchar
DESCRIPTION	varchar

VERSION	
VERSION	varchar

Figure 4.2: BigCloneBench Structure

Table 4.1: BigCloneBench Data Summary

Folder	Functionality	No. of Source Files	LOC	True Clone Pairs						False Clone Pairs
				T1	T2	VST3	ST3	MT3	WT3+4	
2	Download From Web	10,372	1,984,327	1,553	9	22	1,412	2,715	410,611	38,838
3	Secure Hash	4,600	812,629	632	587	525	2,760	24,923	871,717	4,564
4	Copy File	22,113	4,676,552	13,805	3,116	1,210	4,666	24,199	4,725,438	204,108
5	Decompress zip archive	56	3,527	0	0	0	0	1	34	56
6	Connect to FTP Server	472	83,068	9	0	14	50	191	49,161	4,202
7	Bubble Sort Array	1,037	299,525	43	4	21	212	1,752	13,538	5,432
8	Execute update and rollback	131	18,527	3	7	5	0	2	259	78
9	Initialize Java Eclipse Project	669	107,832	0	0	0	0	0	55	1,272
10	Setup SGV Event Handler	1,014	286,416	0	0	285	925	0	245	0
11	Setup SGV	64	6,736	122	10	1	6	97	8,828	24
Total		40,528	8,279,139	16,185	3,787	2,083	10,031	55,106	6,158,975	262,465

Table 4.2: Amount of Training Data

True Clone Pairs (22,663)				False Clone Pairs
T1	T2	VS3	ST3	22,663
13,750	3,104	1,207	4,602	

Table 4.3: Amount of Testing Data

True Clone Pairs (4,724)				False Clone Pairs
T1	T2	VS3	ST3	18,896
2,383	557	307	1,477	

Creating Training data

We followed the methods of creating code clone training data from BCB in the study of Li et al. [33]. We select the true clone pairs in folder#4 contains the largest amount of clone pairs including almost five million clone pairs. Similar to Li et al.'s study, we used clone data from folder#4 to train the model. The criteria are as follows:

1. Clone type 1, 2, and 3
2. Clone type 3 must be only very strong type 3 which has similarity in range [0.9,1) or strongly type 3 which has similarity in range [0.7,0.9)
3. Clone pairs must have gap between start line and end line more than 6 lines

The total number of clone pairs is 22,663 pairs. For false clone or non-clone, the total of false clone pairs in folder#4 is 197,354 pairs. We randomly pick those pairs to be training data in an equal amount as true clones (22,663 pairs). The detail of the training data is shown in Table 4.2.

Creating Testing data

Again, we followed the method in Li et al. [33] to create testing data. We used all folder, except the folder#4 which has already been used for creating the training set, as the test set. The total number of clone pairs is 1,531,000 pairs. For cloned code, we randomly selected clones based on the proportion of each clone type in the BCB data using stratified sampling. Stratified sampling is the method to select the data from the proportion in each category group [38]. For false clones or non-clones, we randomly selected the clone pairs to account for 80 percent of the test data. We select cloned pairs and non-cloned pairs at the ratio of 20% to 80% respectively because it reflects the real-world statistics of clones in software projects [4]. The detail of the test data is shown in Table 4.3.

4.1.2 Code Metrics Extraction

We use code metrics as features to train machine learning models to detect clones. The metrics are divided into syntactic code metrics and semantic code metrics.

Syntactic Code Metrics

Inspired by the previous work by Koschke et al. [30] and Arammongkolvichai et al. [32] , we invented 12 syntactic code metrics to be features of the machine learning model, as shown in Table 4.4. Syntactic code metrics design to gather different characteristics of a method pair. For instance, Number of Token (TokenNo) metric shows the number of total tokens in each clone pair, File name similarity score (SimilarFileNameScore) metric shows either the method occur in the same file or in different files but with the same file names, or in files with totally different file, or Diffience of line of code (DiffLOC) shows how much line of code are difference in each clone pair.

Table 4.4: Syntactic Code Metrics

No.	Metric	Description
1	TokenNo	Difference of number of tokens
2	UniqueTokenNo	Difference of number of unique tokens
3	IdentifierNo	Difference of number of Identifiers
4	UniqueIdentifierNo	Difference of number of unique Identifiers
5	OperatorNo	Difference of number of operators
6	UniqueOperatorNo	Difference of number of unique operators
7	TokenTypesDiversity	Difference of number of unique token types
8	SimilarFileNameScore	File names similarity score
9	SimilarMethodNameScore	Method names similarity score
10	SimilarReturnType	Same return type or not
11	DiffLOC	Difference of lines of code

Semantic Metrics

We applied vector from code2vec [27] to be parts of machine learning features to detect challenging clones by method semantic. First, we extract methods to Java files from source code using Java Parser then give that Java files to code2vec for predicting the semantic vector (Figure 4.2).

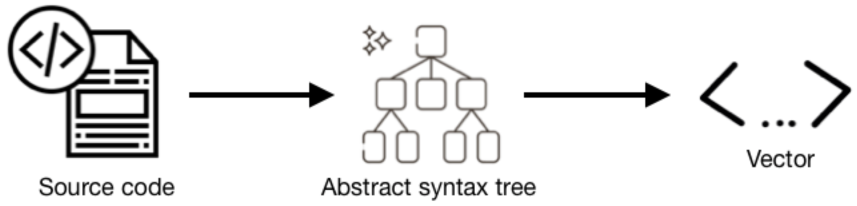


Figure 4.3: code2Vec Process

4.1.3 ML Model Selection

On the training part, we train the model using Weka as our Machine Learning tool. We selected and trained 3 models including Decision Tree (REPTree), Random Forest, and Support Vector Machine (SVM) using Sequential Minimal Optimization (SMO) by using metrics that extract from 22K of true clone pairs and 22k of false clone pairs from BCB as training data.

Decision Tree (REPTree)

REPTree algorithm is an algorithm that uses information gain from data set features as the splitting criteria to construct a decision tree as an example shown in Figure 4.4, and prunes it with reduced error pruning. we use the Weka REPTree default setting of this model which are minimum number of instances per leaf equal to 2 and an unlimited maximum depth of tree to train our model [39]. We decide to use the Decision Tree as our first machine learning model because it is the simplest classification model and much easier to understand than the other classification algorithms.

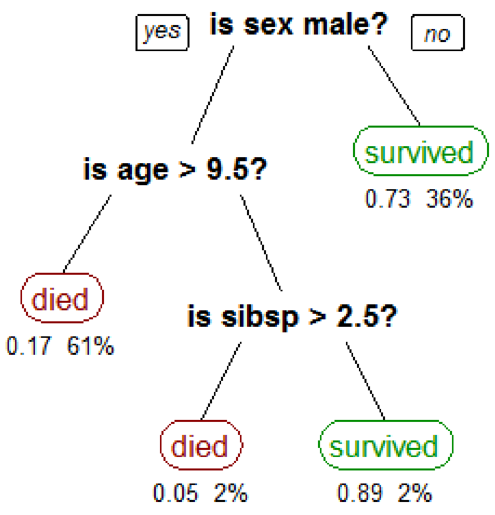


Figure 4.4: An Example of a Decision Tree [1]

Random Forest

Random Forest algorithm is an algorithm that builds several decision trees from random data samples and then gets the prediction from each of them before eventually selects the optimal result by voting. In figure 4.5 illustrates how the random forest algorithm work starting from generating several decision trees from data. Each tree predicts its result before using the majority voting for the final result. The Weka Random Forest default setting of this algorithm that we use to train our model is using 100 trees in the random forest, unlimited maximum depth of tree to train, a minimum number of instances per leaf is equal to 1, and use 1 as the seed of random generator [40]. This is the second machine learning algorithm that we select to use in our tools because it can manage the missing values and maintain a significant portion of the data accuracy. In addition, this algorithm worked well on previous software engineering research [41].

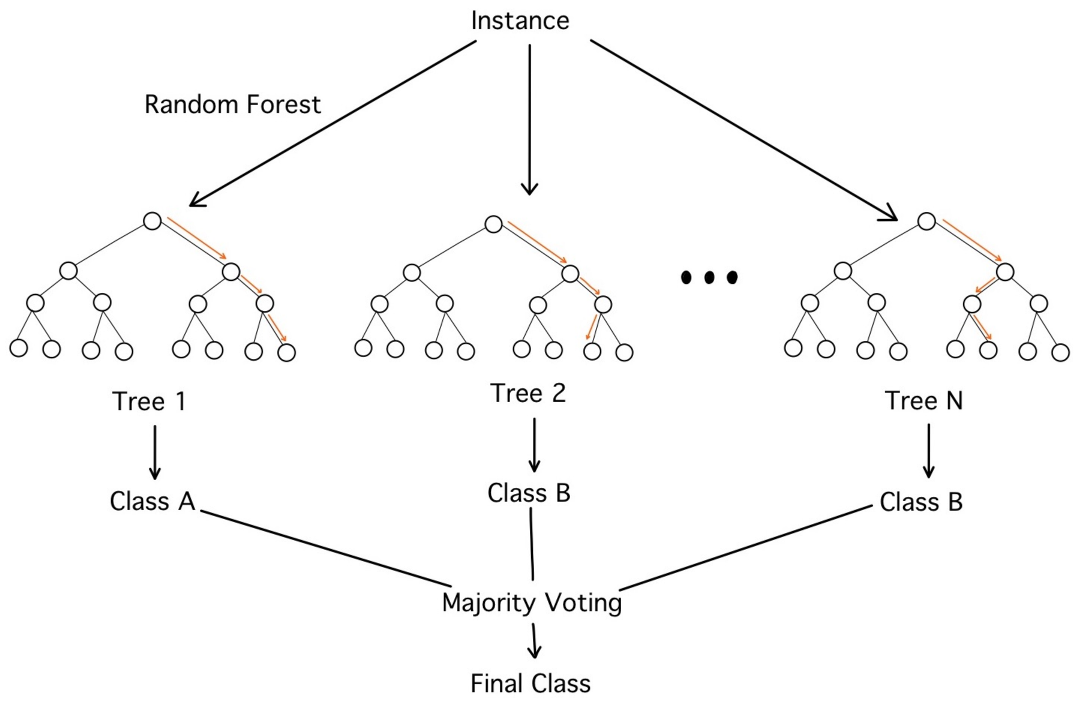


Figure 4.5: Random Forest Simplified

Support Vector Machine (SVM)

Support Vector Machines (SVM) is a supervised learning models in machine learning, that mostly use to solve binary classification problems by using its complex classification algorithms. Figure 4.6 demonstrates how SVM works with an example. Let's the black dot and white dot in the figure 4.6 are 2 kinds of data with labels.The SVM classification algorithms will calculate the best hyperplane (the best line in this 2D graph) that best separate each label of data which is H3 in figure 4.6 and use this hyperplane as the decision boundary to classify data. we select this machine learning model to be one of our model because it is the model that work well on 2-group classification problem which is match with our tool to classify clones and non-clones in the software project.

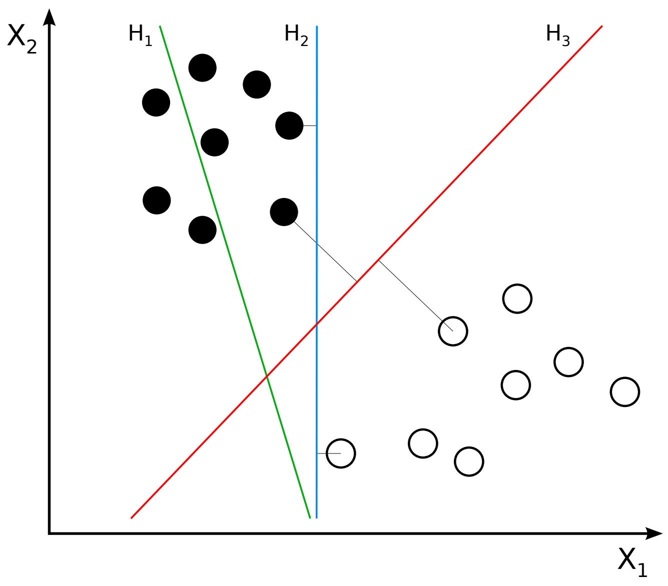


Figure 4.6: How SVM classify the data [2]

Support Vector Machine using Sequential Minimal Optimization (SVM using SMO)

Because SVM training is extremely complex, requires complicate quadratic programming, and takes time. So, we use Sequential Minimal Optimization (SMO) which is the widely used algorithm to solve the complexity and reducing training time during the training Support Vector Machine model to help us calibrate options that use to train the Support Vector Machine model [42]. We decide to include this machine learning classification algorithms to improve the performance and training time of SVM that can capture very complex relationships between datapoint.

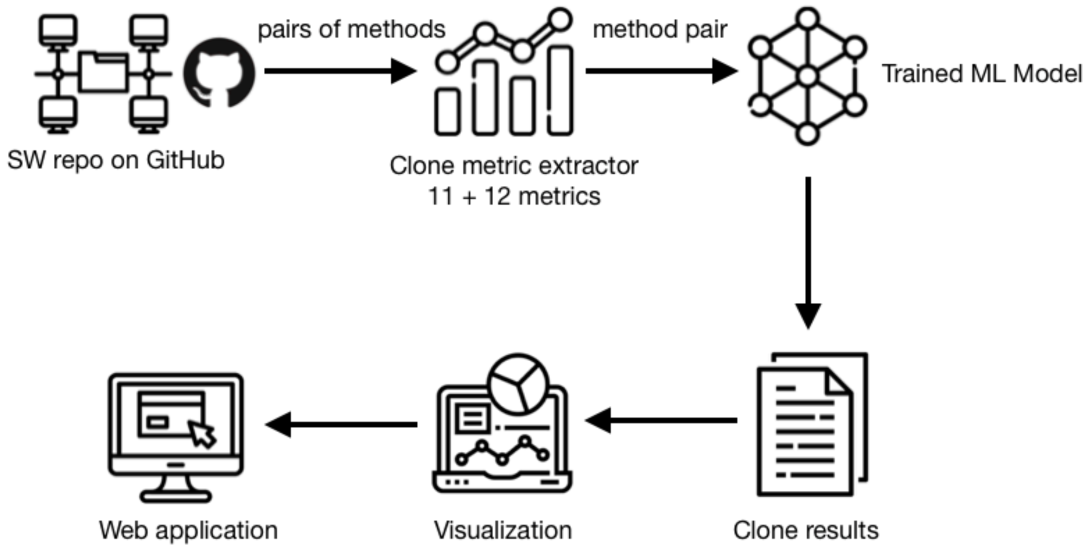


Figure 4.7: Detection process

4.2 Using Trained ML Models for Clone Detection

Figure 4.7 shows the step of how Merry detect clones in a software project. In this detection step, we parse methods from the input software project then pair all methods wisely ($\frac{n\times(n-1)}{2}$ pairs) before extract both syntactic and semantic metrics from each pair to predict which a pair is clone or not by the trained model. When executing Merry in the detection mode, the user can select whether to use both syntactic and semantic metrics, only syntactic metrics, or only semantic metrics. Moreover, users can also select which model will be used on that execution from our 3 trained models. After Merry engine detected clones, it writes the clone results directly to the database (i.e., MongoDB) for visualizing the result nicely and easy to understand on the Merry web application.

4.3 Building a Web application

The creation of web application for improving user experience consists of 2 parts which are front-end and back-end.

4.3.1 Front-end

Hypertext Markup Language (HTML) is the language that we used to build the Merry web application. CSS and Bootstrap also applied to the web application for making the website more beautiful and user friendly. In addition, we also have jQuery that allows the website to receive data and send it back-end via fetch.

Listing 4.1: Connection with GitHub

```
  <a class="nav-link"
  href="https://github.com/login/oauth/authorize?client_id=[CLIENT_ID]
  &redirect_uri=[REDIRECT_URI]/oauth/redirect">Login with GitHub</a>
```

From Listing 4.1, This source code is the source code that redirects the user to login with their GitHub account by using OAuth URI for authorization through GitHub API.

4.3.2 Back-end

Node.js was applied in back-end part by using 2 modules. The first one is **Express** [43] which is used for sending and receiving data between the front-end and the back-end. Another module is **child_process** for executing the Merry engine executable and execute the other necessary commands on the website. Moreover, it connects with GitHub via GitHub API and reads/writes data to/from MongoDB.

GitHub API

Merry web application connects with GitHub to retrieve the user's information, such as authentication and repository. We use the GitHub OAuth app, which is the service that is provided by GitHub. The OAuth stands for an open-standard authorization protocol. It usually uses for a third-party website to grant user information or access from the user without making users directly login on the third-party website.[44] GitHub user can create their OAuth app for using information from GitHub on their website.

Listing 4.2: Connection with GitHub

```
// The req.query object has the query params that
// were sent to this route. We want the code param
const requestToken = req.query.code
axios({
  // make a POST request
  method: 'post',
  // to the Github authentication API, with the client ID, client secret
  // and request token
  url: "https://github.com/login/oauth/access_token?client_id=${clientID}
  &client_secret=${clientSecret}&code=${requestToken}",
  // Set the content type header, so that we get the response in JSOn
  headers: {
       accept: 'application/json'
  }
}).then((response) => {
  // Once we get the response, extract the access token from
  // the response body
  accessToken= response.data.access_token;
  //const username = response.data.username
  // redirect the user to the welcome page, along with the access token
  res.redirect(`/welcome.html?access_token=${accessToken}`)
})
```

From Listing 4.2, This is an example source code of connection with GitHub in Node.js. After login with the GitHub username and password, GitHub will respond to the request token back to the Merry server. Then, the Merry server will use the client ID and the client secret of the GitHub OAuth app combined with the request token to get the access token. The access token is a key that can be subsequently used to request data from GitHub.

MongoDB

Merry web application uses MongoDB to store the results in a database and display the results to the user. The MongoDB stores 2 collections. The first one is `analyedRepoInfo` which contains the username, repository's name of the user, and date/time when the user detects clones. Another one is `result`, which consists of the detection detail of clone pairs. The detail of those 2 collections is shown in Figure 4.8

Listing 4.3: Write data to MongoDB

```
MongoClient.connect(dburl, {
  useNewUrlParser: true,
  useUnifiedTopology: true}, function(err, db)
  {
      if (err) throw err;
      var dbo = db.db("MerryDB");
      dbo.collection("analysedRepoInfo").insertOne(obj, function(err, res) {
          if (err) throw err;
          console.log("document inserted");
      db.close();
      });
  });
```

From Listing 4.3, This is the source code that write data to MongoDB. First, It connects to the database using the URL of MongoDB. Then, data name "obj" is inserted to the collection name "analysedRepoInfo" within the database name "MerryDB".

Listing 4.4: Read data from MongoDB

```
MongoClient.connect(dburl, {
  useUnifiedTopology: true,
  useNewUrlParser: true}, function(err, db)
  {
      if (err) throw err;
      var dbo = db.db("MerryDB");
      //Sort the result by latest date:
      dbo.collection('analysedRepoInfo').aggregate([
     { $sort : { Date : -1 } },
     { $limit: 1 },
     { $project: { _id: { $toString: "$_id" }}},
     { $lookup:
     {
      from: 'Result',
      localField: '_id',
      foreignField: 'ExecutionID',
      as: 'cloneResult'
     }
   } ]).toArray(function(err, result)
  {
      if (err) throw err;
      console.log(JSON.stringify(result));
      res.json(result)
      db.close();
  });
});
```

From Listing 4.4, This is the source code that read the latest data from MongoDB. First, this source code will sort the latest result using command sort and limit. Then, toString will change the type of id from objectid to string. Next, a lookup will search id that matches with ExecutionID.

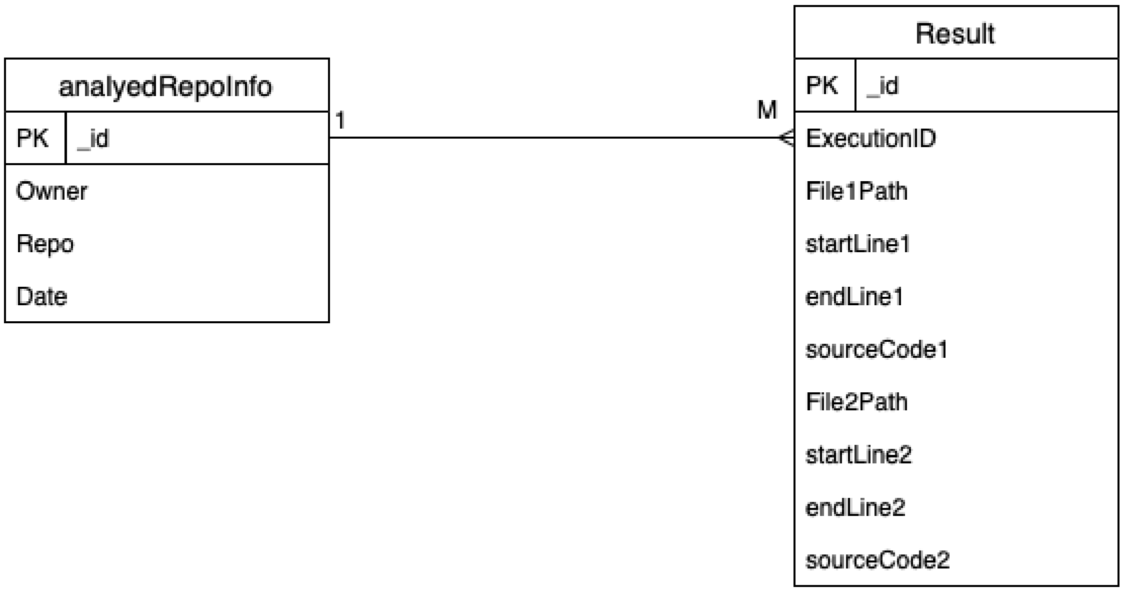


Figure 4.8: Structure of the two collections in the MongoDB database

User Interface of the Merry Web Applications

The Merry web applications consists of 6 pages as follows:

1. **Home page** (see Figure 4.9) contain menu bar, title, and URL form. A menu bar contain with Merry's logo on the left and Login with GitHub button on the right. A URL form will get GitHub URL for users that don't want to keep information in system log.

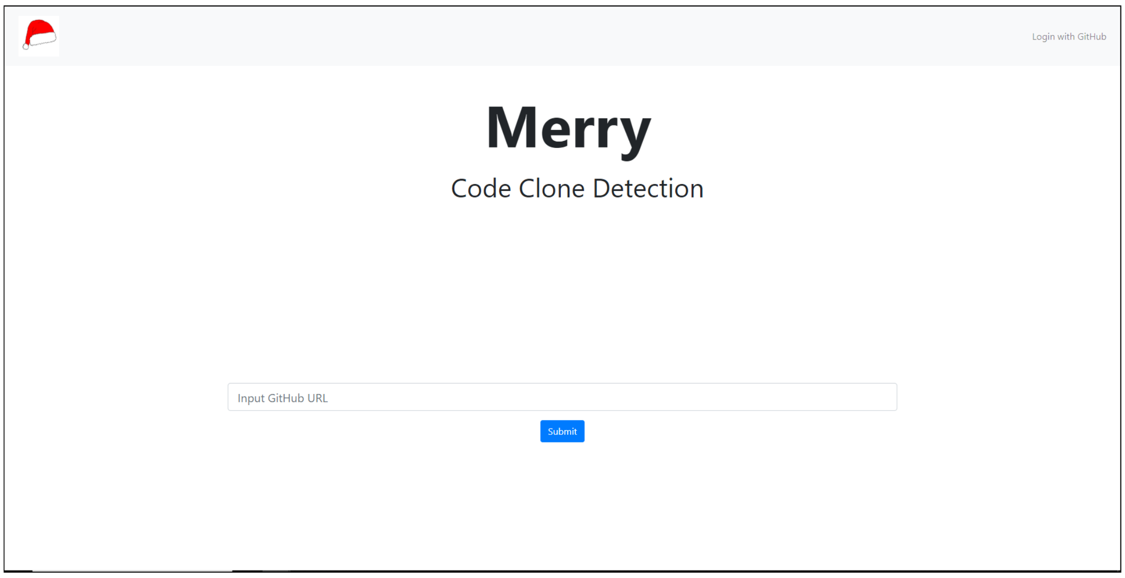
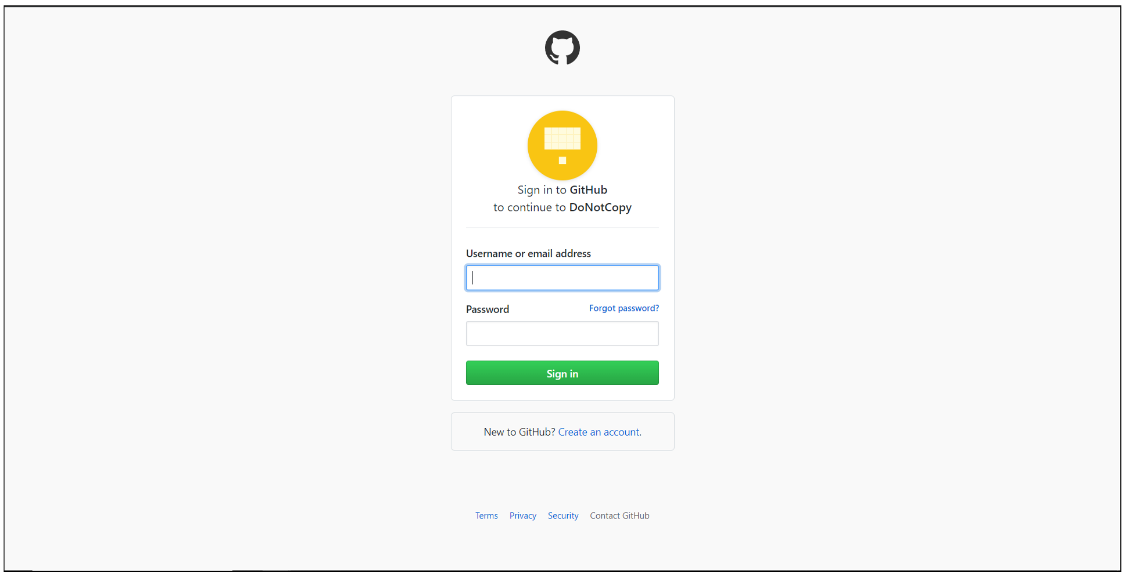


Figure 4.9: Home page

2. **Login page** (Figure 4.10) will come up when user click 'Login with GitHub' button on the right. It come from GitHub API that connect with Merry system.


Sign in to GitHub
to continue to DoNotCopy

Username or email address

Password Forgot password?

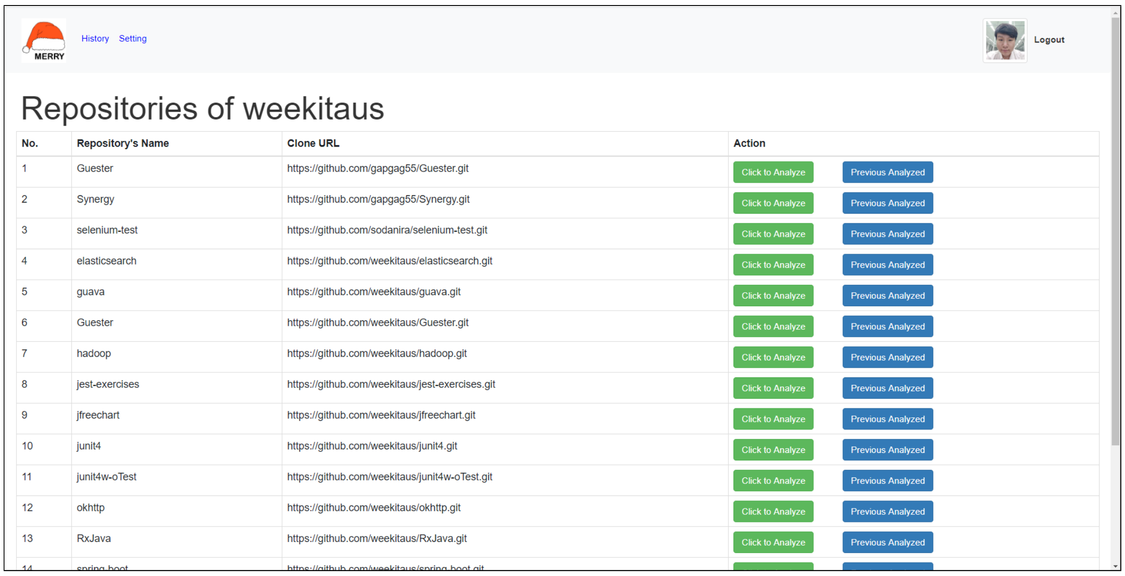
Sign in

New to GitHub? Create an account.

Terms Privacy Security Contact GitHub

Figure 4.10: Login page

3. **Repos page** (Figure 4.11) will appear when users completed the log in process. The repos from GitHub of user account will ready to detect clones.

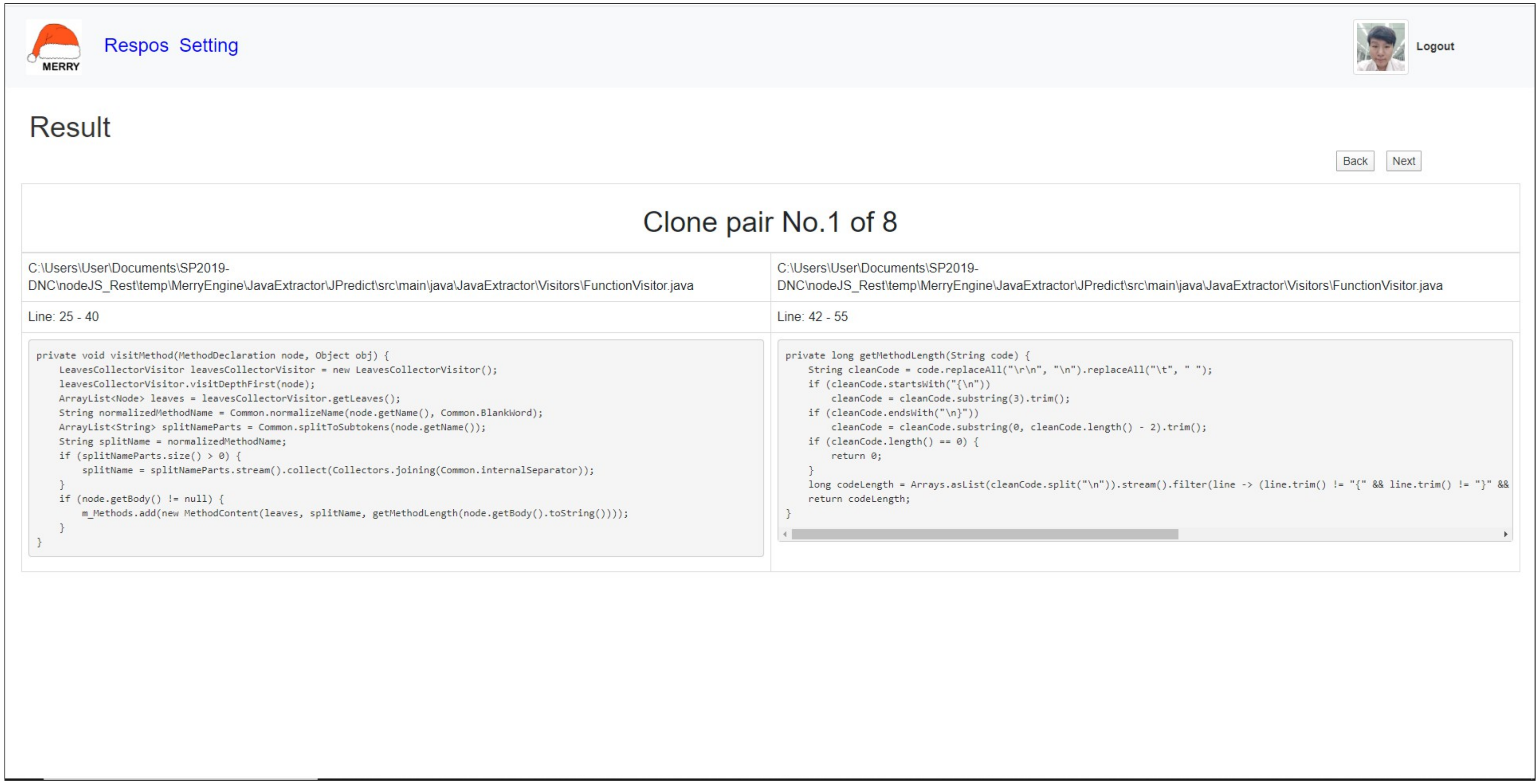


Repositories of weekitaus

No.	Repository's Name	Clone URL	Action	
1	Guester	https://github.com/gapgag55/Guester.git	Click to Analyze	Previous Analyzed
2	Synergy	https://github.com/gapgag55/Synergy.git	Click to Analyze	Previous Analyzed
3	selenium-test	https://github.com/sodanira/selenium-test.git	Click to Analyze	Previous Analyzed
4	elasticsearch	https://github.com/weekitaus/elasticsearch.git	Click to Analyze	Previous Analyzed
5	guava	https://github.com/weekitaus/guava.git	Click to Analyze	Previous Analyzed
6	Guester	https://github.com/weekitaus/Guester.git	Click to Analyze	Previous Analyzed
7	hadoop	https://github.com/weekitaus/hadoop.git	Click to Analyze	Previous Analyzed
8	jest-exercises	https://github.com/weekitaus/jest-exercises.git	Click to Analyze	Previous Analyzed
9	jfreechart	https://github.com/weekitaus/jfreechart.git	Click to Analyze	Previous Analyzed
10	junit4	https://github.com/weekitaus/junit4.git	Click to Analyze	Previous Analyzed
11	junit4w-oTest	https://github.com/weekitaus/junit4w-oTest.git	Click to Analyze	Previous Analyzed
12	okhttp	https://github.com/weekitaus/okhttp.git	Click to Analyze	Previous Analyzed
13	RxJava	https://github.com/weekitaus/RxJava.git	Click to Analyze	Previous Analyzed
14	spring-boot	https://github.com/weekitaus/spring-boot.git		

Figure 4.11: Repo page

4. **Result page**(Figure 4.12) shows all clone pairs that Merry can detect from user's repos when user click the 'Click to Analyze'(green button).



Result

Clone pair No.1 of 8

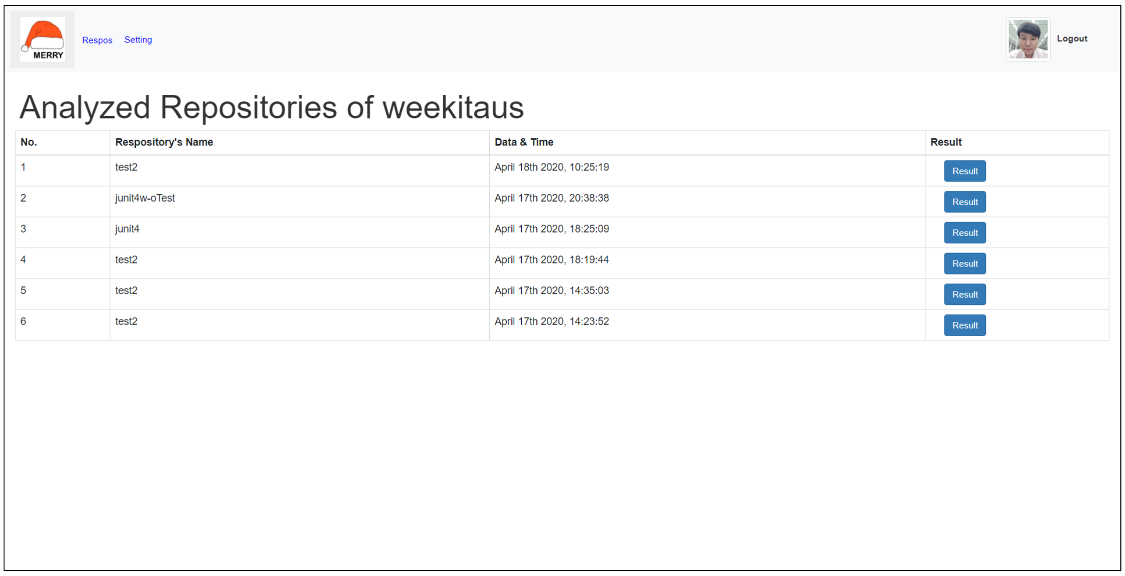
C:\Users\User\Documents\SP2019-DNC\nodeJS_Rest\temp\MerryEngine\JavaExtractor\JPredict\src\main\java\JavaExtractor\Visitors\FunctionVisitor.java	C:\Users\User\Documents\SP2019-DNC\nodeJS_Rest\temp\MerryEngine\JavaExtractor\JPredict\src\main\java\JavaExtractor\Visitors\FunctionVisitor.java
Line: 25 - 40	Line: 42 - 55

```
private void visitMethod(MethodDeclaration node, Object obj) {
    LeavesCollectorVisitor leavesCollectorVisitor = new LeavesCollectorVisitor();
    leavesCollectorVisitor.visitDepthFirst(node);
    ArrayList<Node> leaves = leavesCollectorVisitor.getLeaves();
    String normalizedMethodName = Common.normalizeName(node.getName(), Common.BlankWord);
    ArrayList<String> splitNameParts = Common.splitToSubtokens(node.getName());
    String splitName = normalizedMethodName;
    if (splitNameParts.size() > 0) {
        splitName = splitNameParts.stream().collect(Collectors.joining(Common.internalSeparator));
    }
    if (node.getBody() != null) {
        m_Methods.add(new MethodContent(leaves, splitName, getMethodLength(node.getBody().toString())));
    }
}
```

```
private long getMethodLength(String code) {
    String cleanCode = code.replaceAll("\r\n", "\n").replaceAll("\t", " ");
    if (cleanCode.startsWith("{\n"))
        cleanCode = cleanCode.substring(3).trim();
    if (cleanCode.endsWith("\n}"))
        cleanCode = cleanCode.substring(0, cleanCode.length() - 2).trim();
    if (cleanCode.length() == 0) {
        return 0;
    }
    long codeLength = Arrays.asList(cleanCode.split("\n")).stream().filter(line -> (line.trim() != "{" && line.trim() != "}" &&
    return codeLength;
}
```

Figure 4.12: Result page

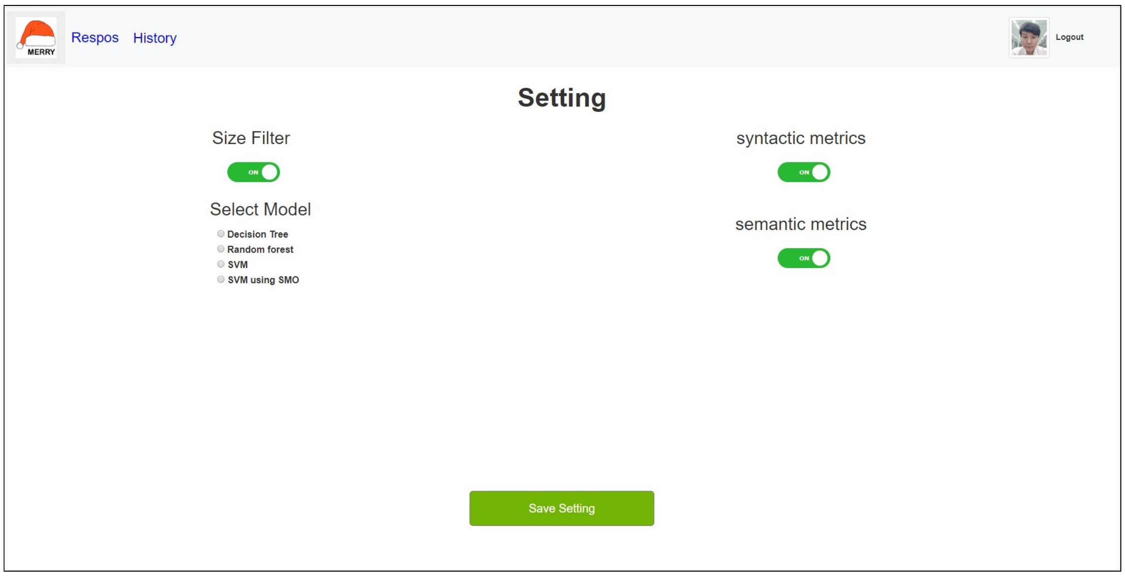
5. **History page** (Figure 4.13) contain result, date and time, and repository name that user has been analyzed.



Analyzed Repositories of weekitaus

No.	Respository's Name	Data & Time	Result
1	test2	April 18th 2020, 10:25:19	Result
2	junit4w-oTest	April 17th 2020, 20:38:38	Result
3	junit4	April 17th 2020, 18:25:09	Result
4	test2	April 17th 2020, 18:19:44	Result
5	test2	April 17th 2020, 14:35:03	Result
6	test2	April 17th 2020, 14:23:52	Result

Figure 4.13: History page



Setting

Size Filter: ON

Select Model
- Decision Tree
- Random forest
- SVM
- SVM using SMO

syntactic metrics: ON

semantic metrics: ON

Figure 4.14: Setting page

6. **Setting page** (Figure 4.14) uses for setting the features of Merry engine including size filter, syntactic metrics, and semantic metrics. All of features be able to select on or off.

CHAPTER 5
EVALUATION RESULTS

In this chapter, we discuss the results of our experiments to evaluate the performance of the Merry Clone Detection Tool.

5.1 Evaluation of Merry Clone Detection Engine using BCB

This section shows the evaluation of the Merry clone detection tool on data in the BCB database, which is the largest clone ground-truth on Java language that contains around 6 million pairs of true clone. Evaluating this ground-truth makes us be able to compute precision, recall, and f1-score of our approach.

5.1.1 Methodology

To evaluate our Merry engine, we use the test data containing 4,724 true clone pairs and 18,896 false clone pairs from all folders in BCB except folder#4 that we already used as the training data to train the model. The reason we use this amount of clone pairs is in real software projects usually have clones around 20 percent [4], so we decided to make the testing data similar to real software projects.

Table 5.1: Merry execution parameters

Parameters	Values	Description
-model	decisiontree	Use Decision Tree as the model on the execution.
	randomforest	Use Random Forest as the model on the execution.
	svm	Use Support Vector Machine as the model on the execution.
	smo	Use Support Vector Machine using Sequential Minimal Optimization as the model on the execution.
-syn	on	Use syntactic metrics as machine learning features on the execution.
	off	Don't use syntactic metrics as machine learning features on the execution.
-sem	on	Use semantic metrics as machine learning features on the execution.
	off	Don't use semantic metrics as machine learning features on the execution.

Table 5.1 shows the list of Merry's parameters that we use to run our experiments with values and description of each parameter.

5.1.2 Error Measures

We use the well-known error measures in the information retrieval field which are precision, recall, and F1-score to measure Merry clone detection tool. For easier to understand these measurement, Figure 5.1 demonstrates 4 importance terms as follows:

- True positive: the elements that actually are true and predicted true
- False positive: the elements that actually are false but predicted as true.
- False negative: the elements that actually are true but predicted as false.
- True negative: the elements that actually are false and predicted false.

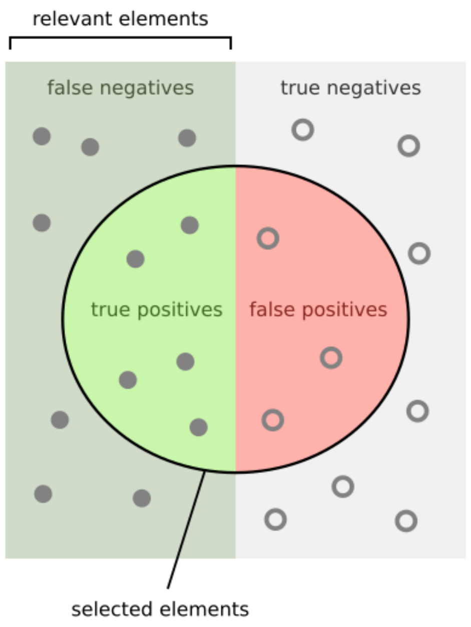


Figure 5.1: Terminology of the confusion matrix [3]

Precision measures the percentage of correctness from elements that was predicted true as show in Figure 5.2. It can be calculated by using this equation.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall measures the percentage of completeness from all elements that are actually true as shown in Figure 5.2. It can be calculated by using this equation.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

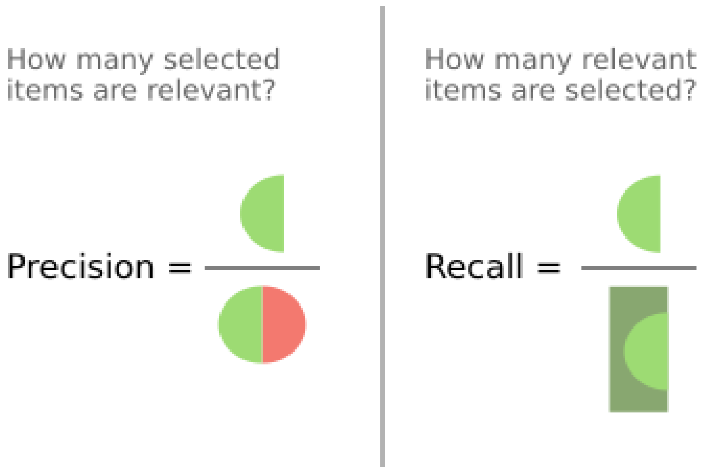

How many selected items are relevant?
How many relevant items are selected?
Precision =
Recall =

Figure 5.2: Precision and Recall measurement [3]

F1-Score measures a harmonic mean (average) of Precision and Recall. It can be calculated by using this equation.

$$F1\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

5.1.3 Evaluation Results

Table 5.2 shows the result of our experimental testing by comparing Precision, Recall, and F1-Score of each model on three different combinations of metrics (syntactic and semantic, only syntactic, only semantic) with the baseline of Randomization model (a simple model that randomly classifies a pair of code as cloned or non-cloned).

Figure 5.3 demonstrates the difference of precision when running Merry with Decision Tree Model, Random Forest Model, Support Vector Machine (SVM), and SVM using SMO Model compared with the precision of Randomization model. As the graph show, Merry with any model gives better precision than the randomized model. In addition, we also observe that most of the time, using both syntactic metrics and semantic

Table 5.2: Experimental result on sampled BCB dataset

Model	**Metrics**	**Precision**	**Recall**	**F1-Score**
Randomization (baseline)		0.20	0.49	0.28
Decision Tree	Syntactic + Semantic	0.89	0.86	0.87
	Syntactic	0.95	0.72	0.86
	Semantic	0.68	0.87	0.76
Random Forest	Syntactic + Semantic	0.97	0.86	0.91
	Syntactic	0.97	0.80	0.87
	Semantic	0.70	0.87	0.78
SVM	Syntactic + Semantic	0.97	0.85	0.91
	Syntactic	0.97	0.79	0.87
	Semantic	0.62	0.90	0.73
SVM using SMO	Syntactic + Semantic	0.98	0.89	0.93
	Syntactic	0.97	0.69	0.81
	Semantic	0.63	0.90	0.74

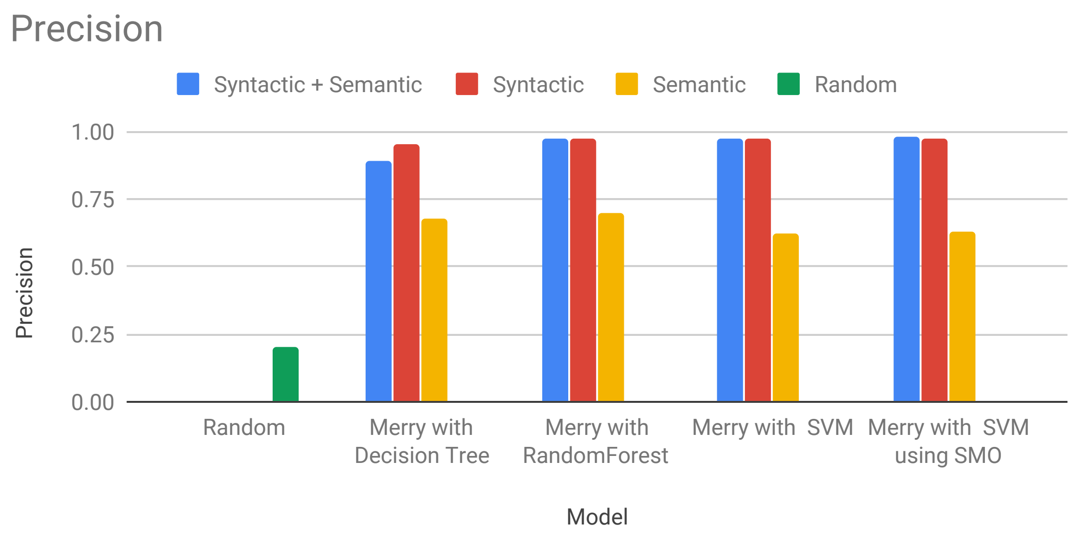


Figure 5.3: Detection process

metrics perform equal to or better than using only syntactic or semantic features for precision.

Figure 5.4 demonstrates the difference of recall when running Merry with Decision Tree Model, Random Forest Model, SVM, and SVM using SMO Models with the recall of Randomization model as the baseline on the testing data. From the graph, Merry, with any model giving better recall than the Randomization model, just like precision. Moreover, we also observe that using both syntactic metrics and semantic metrics perform better than using only syntactic metrics for recall of the result.

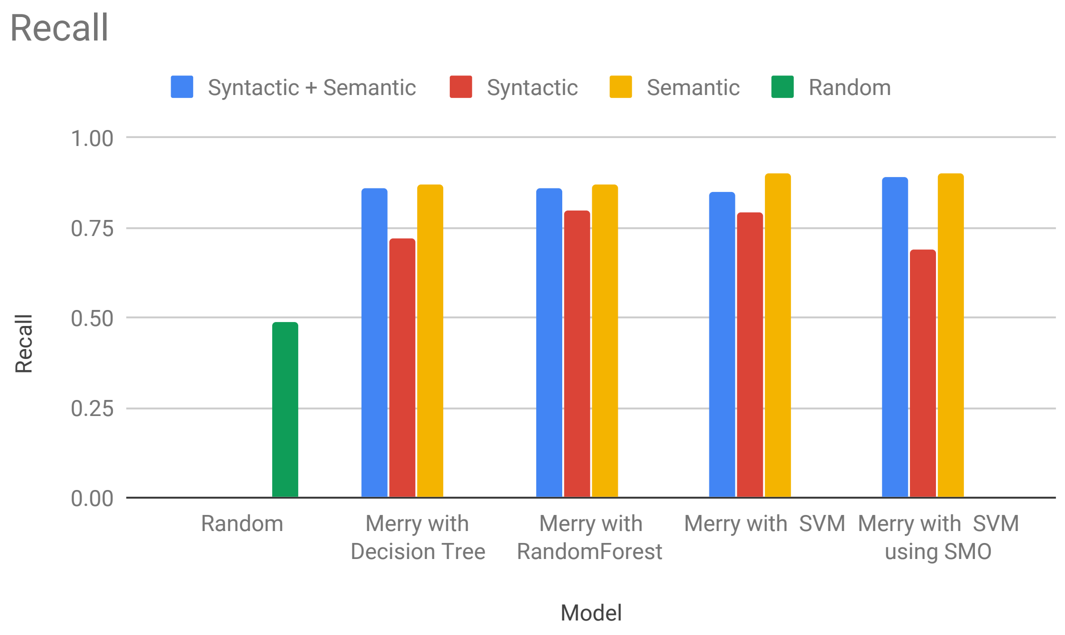


Figure 5.4: Detection process

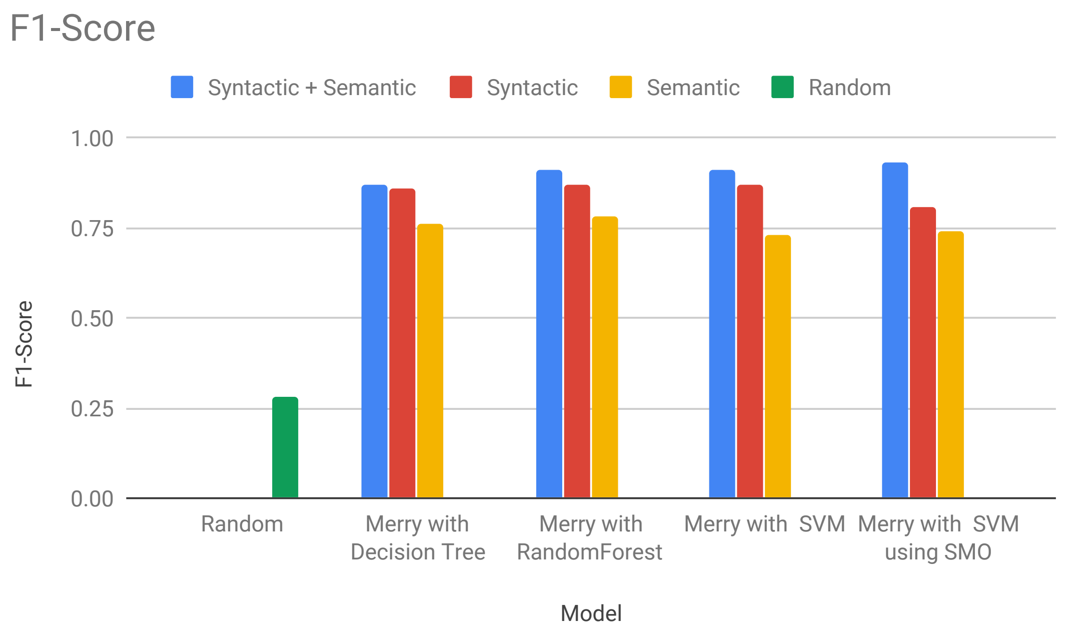


Figure 5.5: Detection process

Figure 5.5 demonstrates the difference of F1-score when running Merry with Decision Tree Model, Random Forest Model, SVM, and SVM using SMO Models with the f1-score of Randomization model as the baseline on the testing data. As shown in the graph, Merry runs with any model giving a better f1-score than the Randomization model, just like precision. Interestingly, we found that using semantic metrics perform the best for the f1-score.

5.1.4 Discussion

We found that Merry running with all the four models performs better than the baseline Randomization model, and the best Merry's model that gives the best result using F1-score is Support Vector Machine using Sequential Minimal Optimization or SVM using SMO. Moreover, using only syntactic metrics not perform well in terms of recall. On the other hand, using only semantic metrics also not perform well in terms of precision.

5.2 Evaluation of Merry Clone Detection Engine on Real Software Projects

For this testing on real software projects, we want to evaluate how effective the Merry clone detection tool when users use it to detect clones in their software project.

5.2.1 Methodology

To evaluation of Merry Clone Detection Engine with SVM using SMO model on Real Software Project we run Merry on 3 projects including the JUnit4 [45] which is the open-source software on GitHub (collected on 17th April 2020), Natty [46] (collected on 17th April 2020), and our project Merry. After the Merry clone detection tool finished running on software projects, we found that the main source code of software projects contain 187 clone pairs. Then we manually validate 187 clone pairs by look into those pairs one by one. Each of the three members in the team decides which pairs are clone or non-clone by each person's decision. After that, we decide if the result is a true clone pair by the majority of the decisions. Natty and Merry were found a little of clone pairs because they are small projects while JUnit was found 187 clone pairs. For the precision measure, on Natty and our Merry clone detection tool performs quite well while on JUnit, we get the precision around 0.4. From our manual validation experience, we found that JUnit contains less non-challenging clones and many challenging clones.

Table 5.3: Merry's result on real software project

Project	LOC	#methods	No. clone pairs	TP	FP	Precision	Runtime(second)
JUnit4	7550	1529	187	77	110	0.41	552
Natty	1761	146	9	7	2	0.77	99
Merry	931	102	8	6	2	0.75	115

5.2.2 Evaluation Result

This section demonstrate the result of Merry clone detection tool when run on real software projects (JUnit4, Natty, and Merry) including the example of challenging clone and false clone from real software projects.

Table 5.3 show the result of real software project that are run on Merry clone detection tool. Natty and Merry were found a little of clone pairs because they are small projects while JUnit was found 187 clone pairs. For the precision measure, on Natty and our Merry clone detection tool performs quite well while on JUnit, we get the precision around 0.4. From our manual validation experience, we found that JUnit contains less non-challenging clones and many challenging clones.

Listing 5.1: Example of challenge true clone

```java
static public Test createTest(Class<?> theClass, String name) {
   Constructor<?> constructor;
   try {
      constructor = getTestConstructor(theClass);
   } catch (NoSuchMethodException e) {
      return warning("Class " + theClass.getName() + " has no public
          ↪ constructor TestCase(String name) or TestCase()");
   }
   Object test;
   try {
      if (constructor.getParameterTypes().length == 0) {
         test = constructor.newInstance(new Object[0]);
         if (test instanceof TestCase) {
            ((TestCase) test).setName(name);
         }
      } else {
         test = constructor.newInstance(new Object[] { name });
      }
   } catch (InstantiationException e) {
      return (warning("Cannot instantiate test case: " + name + " (" +
            ↪ Throwables.getStacktrace(e) + ")"));
    } catch (InvocationTargetException e) {
        return (warning("Exception in constructor: " + name + " (" +
            ↪ Throwables.getStacktrace(e.getTargetException()) + ")"));
    } catch (IllegalAccessException e) {
        return (warning("Cannot access test case: " + name + " (" +
            ↪ Throwables.getStacktrace(e) + ")"));
    }
    return (Test) test;
```

```
private void addTestsFromTestCase(final Class<?> theClass) {
    fName = theClass.getName();
    try {
        getTestConstructor(theClass);
    } catch (NoSuchMethodException e) {
        addTest(warning("Class " + theClass.getName() + " has no public
            ↪ constructor TestCase(String name) or TestCase()"));
        return;
    }
    if (!Modifier.isPublic(theClass.getModifiers())) {
        addTest(warning("Class " + theClass.getName() + " is not public"));
        return;
    }
    Class<?> superClass = theClass;
    List<String> names = new ArrayList<String>();
    while (Test.class.isAssignableFrom(superClass)) {
        for (Method each : MethodSorter.getDeclaredMethods(superClass)) {
            addTestMethod(each, names, theClass);
        }
        superClass = superClass.getSuperclass();
    }
    if (fTests.size() == 0) {
        addTest(warning("No tests found in " + theClass.getName()));
    }
}
```

Listing 5.1 show the example of true clone that is one challenge clone from JUnit4. This true clone pair is challenge because both of them perform pretty same functionality which adding the new test but both of them are syntactically difference and have difference inputs.

Listing 5.2: Example of false clone

```java
protected void runChild(final FrameworkMethod method, RunNotifier notifier) {
    Description description = describeChild(method);
    if (isIgnored(method)) {
        notifier.fireTestIgnored(description);
    } else {
        Statement statement = new Statement() {

            @Override
            public void evaluate() throws Throwable {
                methodBlock(method).evaluate();
            }
        };
        runLeaf(statement, description, notifier);
    }
}
```

```java
public Result run(Runner runner) {
    Result result = new Result();
    RunListener listener = result.createListener();
    notifier.addFirstListener(listener);
    try {
        notifier.fireTestRunStarted(runner.getDescription());
        runner.run(notifier);
        notifier.fireTestRunFinished(result);
    } finally {
        removeListener(listener);
    }
    return result;
}
```

Listing 5.2 show the example of false clone that report from JUnit4 [45]. We think that this false clone was reported because both of the source have almost similar method name and almost same amount line of code.

5.3 Evaluation of Merry Web Application by Users

This evaluation is the user testing to evaluate that the Merry clone detection tool that is Web Application based is more convenient than the existing command-line based clone detection tools or not.

5.3.1 Methodology

To evaluate the web application, we have created the Google form for gathering the opinions from users. Moreover, we have created a video that contains the basic concept of code clone and a demonstration of how to use Merry and Simian [8]. Merry is our web-based that provides a user interface. Simian is a command-line based code clone detection tool. There are 2 forms for Merry and Simian [8]. Each participant watches only one video and answers, only one form to avoid bias. There are 7 questions in Google form below:

1. How much experience in programming that you have?
2. Are you familiar with code clones (duplicated code)?
3. How much code clones (duplicated code) is your concern when you develop software?
4. If this Simian/Merry tool will be applied to one of your software project, how likely will you use it?
5. Did you find the Simian/Merry tool easy to understand (how to run, how to interpret results)?
6. Did you find the Simian/Merry tool easy to use (environment needed to run to the tool and see the results)?
7. Any other comments on your impression on the Merry tool?

The questionnaire can be found in APPENDIX.

5.3.2 Evaluation Results

From the result of the questionnaires, we found that users want to evaluate users who evaluate the Merry clone detection tool willing to apply the tool in their software project more than users who evaluate Simian clone detection tool [8] (existing command-line based tool). Moreover, most users who evaluate the Merry clone detection tool evaluate that the tool is easy to use and easy to understand the result. In contrast, users who evaluate the Simian clone detection tool mostly evaluate that the tool is difficult to use and difficult to understand the result. The details are shown in Figure 5.6-Figure 5.11.

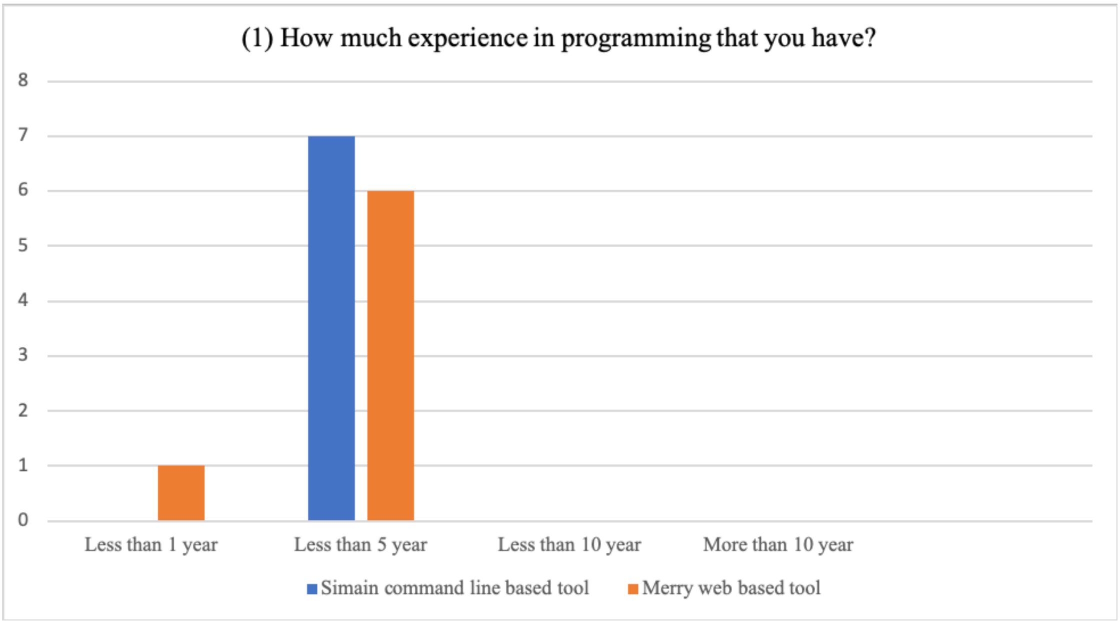


Figure 5.6: Question 1

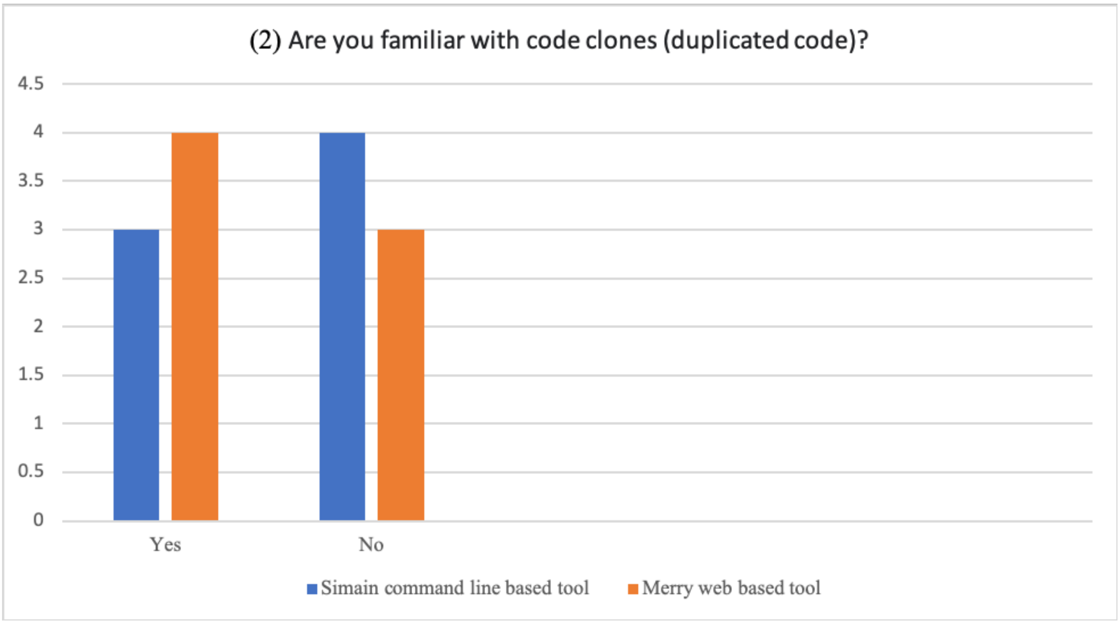


Figure 5.7: Question 2

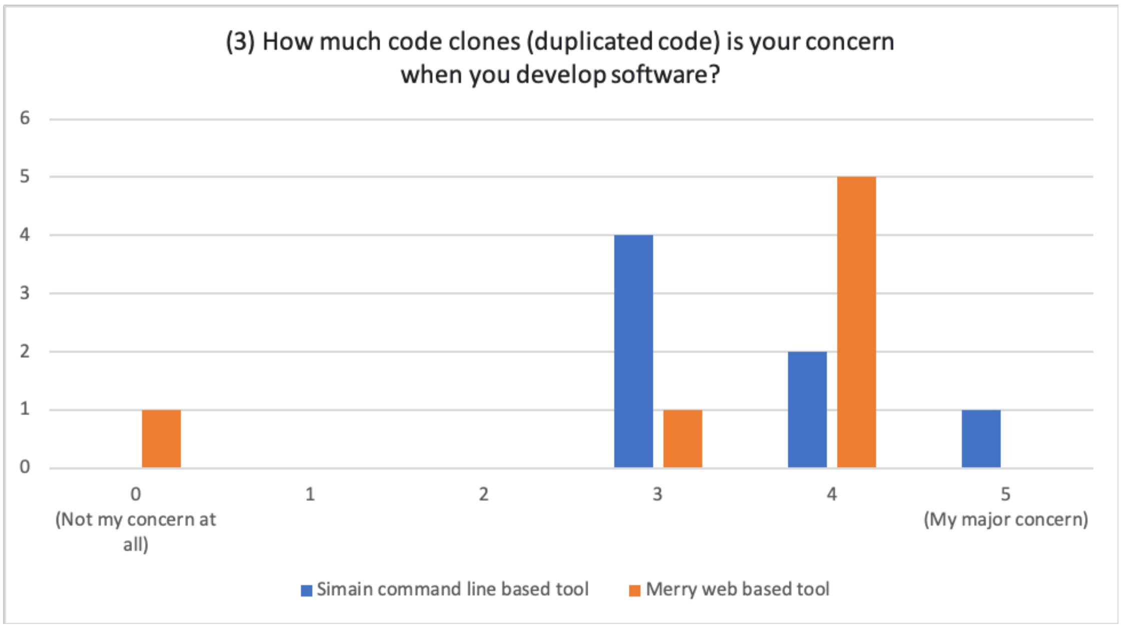


Figure 5.8: Question 3

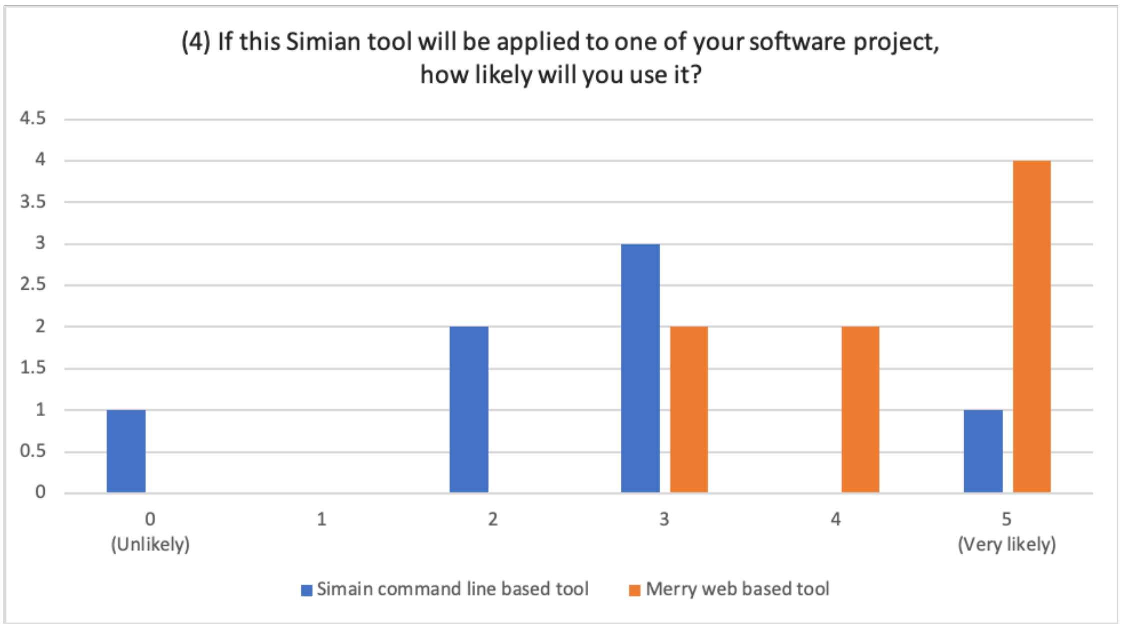


Figure 5.9: Question 4

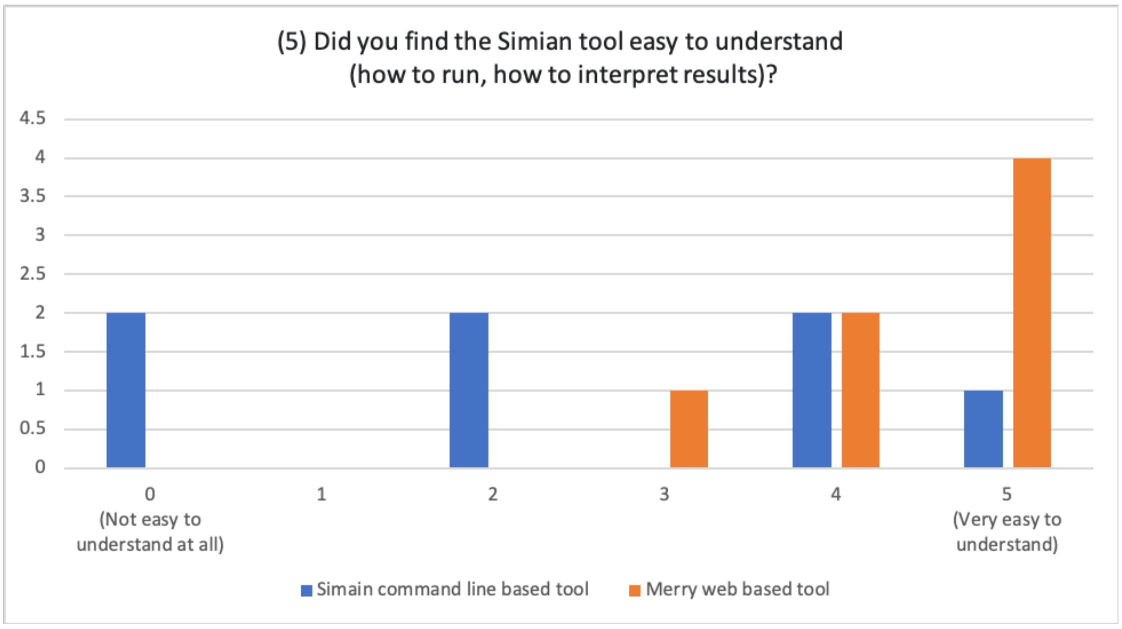


Figure 5.10: Question 5

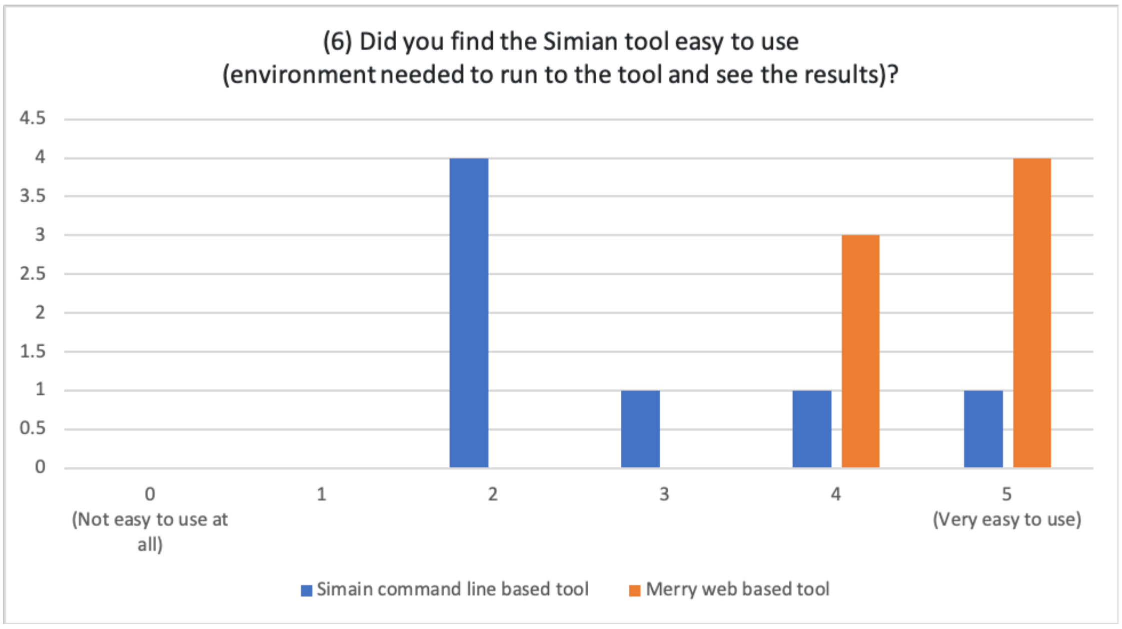


Figure 5.11: Question 6

CHAPTER 6
CONCLUSIONS

This chapter concludes and discusses our project and the results. There are three sections consist of conclusion, problems and limitations, and future work.

6.1 Conclusion

Merry is a web-based code clone detection tool that use machine learning techniques. We aim to accurately detect clones and improve user experience by making Merry tool running on a web application. Merry can be divided into 2 parts. The first part is the Merry engine. We trained 4 machine learning models that are decision tree (REPTree), random forest, support vector machine (SVM), and Support Vector Machine using Sequential Minimal Optimization (SVM using SMO). We evaluated the performance of Merry by using BigCloneBench [34]. The evaluation results show that Merry's clone detection precision, recall, and F1-score are high, which means the Merry engine performs well on the BCB database. Another part is the Merry web application. Merry web application is directly connected with GitHub to make it more user friendly. According to the user study, the Merry web application is found to be easier to use and more convenient than the command line based tool, Simian.

6.2 Problems and limitations

One of the current version limitations is the tool supports only Java language. Even our syntactic and semnatic source code metrics can be able to apply to any language, this version of the Merry clone detection tool includes only a Java parser and a Java tokenizer. Therefore, now it can detect only Java language clones.

Code2vec run-time performance is one of our problems and limitations. Because code2vec is designed to analyze only one source code at a time. Nonetheless, we use it to create semantic metrics for every method that we parse from all java files in the software project, it causes the slow run-time.

The precision and recall that we evaluated Merry on the BCB database might not reflect its performance on real software projects. For example, JUnit4 shows the limitations of the tool. JUnit contains less Type-1 and Type-2 clones that can cause the ratio of the difficult Type-3 clone pairs.

Another limitation is the current version of MongoDB does not support big document size. The maximum of the MongoDB document size is limited at 16 megabytes. It causes the Merry web application not be able to show the visualization of a huge software project because of its massive number of clones that exceeds 16 megabytes.

6.3 Future work

In the future, we plan to improve Merry to be able to detect clones in other languages by importing and implementing the function to be able to parse and tokenize other languages more than just Java.

Second, we aim to improve the code2vec run-time by re-implement it to be able to run in several threads at a time to improve the run-time performance of code2vec for making the Merry clone detection tool run faster.

In terms of precision and recall, we aim to train multiple machine learning models by clone types and use each model to detect each type of clones. This may improve the performance of precision and recall by reducing the false positive in the challenging clones.

To solve the MongoDB limitation, we need to adjust the Merry web application to query a part of MongoDB document at a time (i.e., divide them into pages). This will ensure that document is under 16 megabytes and can show on the visualized page.

REFERENCES

[1] "Decision tree learning", Wikimedia Foundation; Apr 2020, [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning.

[2] Foundation W.. "Support-vector machine";, Online; accessed 18 April 2020, [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine.

[3] Wikimedia. "Precision and recall"; Mar 2020, Online; accessed 18 April 2020, https:// en.wikipedia.org/ wiki/ Precision_and_recallDefinition_(information_retrieval_context).

[4] Roy CK., Cordy JR., Koschke R., "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach", Science of computer programming. 2009;74(7):470–495.

[5] Bellon S., Koschke R., Antoniol G., Krinke J., Merlo E., "Comparison and evaluation of clone detection tools", IEEE Transactions on software engineering;.

[6] Roy CK., Cordy JR., "An empirical study of function clones in open source software", In: 2008 15th Working Conference on Reverse Engineering. IEEE; 2008. p. 81–90.

[7] Roy CK., Cordy JR., "NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization", In: 2008 16th iEEE international conference on program comprehension. IEEE; 2008. p. 172–181.

[8] Simian T., "Simian tool", URL http://www redhillconsulting com au/products/simian;.

[9] Marcus A., Maletic JI., "Identification of high-level concept clones in source code", In: Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001). IEEE; 2001. p. 107–114.

[10] Li Z., Lu S., Myagmar S., Zhou Y., “CP-Miner: Finding copy-paste and related bugs in large-scale software code”, IEEE Transactions on software Engineering. 2006;32(3):176–192.

[11] Göde N., Koschke R., “Incremental clone detection”, In: 2009 13th European Conference on Software Maintenance and Reengineering. IEEE; 2009. p. 219–228.

[12] Kamiya T., Kusumoto S., Inoue K., “CCFinder: a multilinguistic token-based code clone detection system for large scale source code”, IEEE Transactions on Software Engineering. 2002;28(7):654–670.

[13] Kamiya T., “The official CCFinderX website”, URL http://www ccfinder net/ ccfinderx html Last accessed November. 2008;.

[14] Baxter ID., Yahin A., Moura L., Sant’Anna M., Bier L., “Clone detection using abstract syntax trees”, In: Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272). IEEE; 1998. p. 368–377.

[15] SimScan T., “URL http://www. blue-edge. bg/simscan”, Last accessed November. 2008;.

[16] Jiang L., Misherghi G., Su Z., Glondu S., “Deckard: Scalable and accurate tree-based detection of code clones”, In: 29th International Conference on Software Engineering (ICSE’07). IEEE; 2007. p. 96–105.

[17] Davey N., Barson P., Field S., Frank R., Tansley D., “The development of a software clone detector”, International Journal of Applied Software Technology. 1995;.

[18] Krinke J., “Identifying similar code with program dependence graphs”, In: Proceedings Eighth Working Conference on Reverse Engineering. IEEE; 2001. p. 301–309.

[19] Gabel M., Jiang L., Su Z., “Scalable detection of semantic clones”, In: Proceedings of the 30th international conference on Software engineering; 2008. p. 321–330.

[20] Komondoor R., Horwitz S., “Using slicing to identify duplication in source code”, In: International static analysis symposium. Springer; 2001. p. 40–56.

[21] Rouse M.. “What is Web Application (Web Apps) and its Benefits”, TechTarget; 2019, [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app.

[22] Conallen J., “Modeling Web application architectures with UML”, Communications of the ACM. 1999;42(10):63–70.

[23] “GitHub”; August 2019 [cited 10 November 2019], [Online]. Available: https://github.com/.

[24] “The most popular database for modern apps”. MongoDB [cited 1 April 2020]; [Online]. Available: https://www.mongodb.com/.

[25] “WEKA”. Weka 3 - Data Mining with Open Source Machine Learning Software in Java [cited 1 April 2020];[Online]. Available: https://www.cs.waikato.ac.nz/ ml/ weka/.

[26] “JavaParser”. JavaParser [cited 1 April 2020]; [Online]. Available: http://javaparser.org/.

[27] Alon U., Zilberstein M., Levy O., Yahav E., “code2vec: Learning distributed representations of code”, Proceedings of the ACM on Programming Languages. 2019;3(POPL):1–29.

[28] “CODE2VEC”. code2vec [cited 1 April 2020]; [Online]. Available: https://code2vec.org/.

[29] “Metric Definitions | SonarQube Docs”; August 2019 [cited 11 November 2019], [Online]. Available: https://docs.sonarqube.org/latest/user-guide/metric-definitions/.

[30] Koschke R., Bazrafshan S., “Software-clone rates in open-source programs written in C or C++”, In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). vol. 3. IEEE; 2016. p. 1–7.

[31] Saini V., Farmahinifarahani F., Lu Y., Baldi P., Lopes CV., “Oreo: Detection of clones in the twilight zone”, In: Proceedings of the 2018 26th ACM Joint Meeting

on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM; 2018. p. 354–365.

[32] Vara A., Rainer K., Chaiyong R., Morakot C., Thanwadee S., “Improving Clone Detection Precision using Machine Learning Techniques”, In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. IWESEP; 2019. .

[33] Li L., Feng H., Zhuang W., Meng N., Ryder B., “CCLearner: A deep learning-based clone detection approach”, In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE; 2017. p. 249–260.

[34] Svajlenko J., Roy CK., “Evaluating clone detection tools with bigclonebench”, In: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE; 2015. p. 131–140.

[35] Sajnani H., Saini V., Svajlenko J., Roy CK., Lopes CV., “SourcererCC: Scaling code clone detection to big-code”, In: Proceedings of the 38th International Conference on Software Engineering; 2016. p. 1157–1168.

[36] Cordy JR., Roy CK., “The NiCad clone detector”, In: 2011 IEEE 19th International Conference on Program Comprehension. IEEE; 2011. p. 219–220.

[37] White M., Tufano M., Vendome C., Poshyvanyk D., “Deep learning code fragments for code clone detection”, In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE; 2016. p. 87–98.

[38] “Stratified sampling”, Wikimedia Foundation; April 2020, [Online]. Available: https://en.wikipedia.org/wiki/Stratified_sampling [cited 1 April 2020].

[39] “REPTree”; December 2019 [cited 1 April 2020], [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/REPTree.html.

[40] “RandomForest”; December 2019 [cited 1 April 2020], [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html.

[41] Choetkiertikul M., Dam HK., Tran T., Ghose A., "Predicting delays in software projects using networked classification", In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE; 2015. p. 353–364.

[42] "RandomForest". SMO. December 2019 [cited 1 April 2020];[Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/SMO.html.

[43] Expressjs. "expressjs/express"; 2020, Online; accessed 1 April 2020, [Online]. Available: https://github.com/expressjs/express.

[44] "OAuth", Wikimedia Foundation; Apr 2020, [Online]. Available: https://en.wikipedia.org/wiki/OAuth.

[45] JUnit. "JUnit4";, Online; accessed 18 April 2020, [Online]. Available: https://junit.org/junit4/.

[46] Joestelmach. "joestelmach/natty";, Online; accessed 18 April 2020, [Online]. Available: https://github.com/joestelmach/natty.

BIOGRAPHIES

NAME	Mr. Vara Arammongkolvichai
DATE OF BIRTH	24 June 1998
PLACE OF BIRTH	Nonthaburi, Thailand
INSTITUTIONS ATTENDED	Saint Gabriel's College, 2015:
	High School Diploma
	Mahidol University, 2020:
	Bachelor of Science (ICT)

NAME	Mr. Weekit Ausavaserenont
DATE OF BIRTH	11 September 1997
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Saint Gabriel's College, 2015:
	High School Diploma
	Mahidol University, 2020:
	Bachelor of Science (ICT)

NAME	Miss. Wannaporn Vichaisri
DATE OF BIRTH	19 June 1998
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Suksanari School, 2015:
	High School Diploma
	Mahidol University, 2020:
	Bachelor of Science (ICT)

APPENDIX A
MANUAL CLONE VALIDATION RESULTS

Junit4 Manual Validation Result

Total = 187 clone pairs True Positive = 77 pairs False Positive = 110 pairs

Clone Pair Number	File Path 1	Start line 1	End line 1	File Path 2	Start line 2	End line 2	Judge 1 (Vara)	Judge 2 (Wannaporn)	Judge 3 (Weekit)
1	junit4/src/main/java/org/junit/experimental/theories/PotentialAssignment.java	17	47	junit4/src/main/java/org/junit/internal/management/ReflectiveThreadMXBean.java	48	69	0	0	0
2	junit4/src/main/java/org/junit/internal/builders/AllDefaultPossibilitiesBuilder.java	27	43	junit4/src/main/java/org/junit/internal/runners/SuiteMethod.java	27	40	0	0	0
3	junit4/src/main/java/org/junit/internal/builders/AllDefaultPossibilitiesBuilder.java	27	43	junit4/src/main/java/org/junit/internal/builders/AnnotatedBuilder.java	80	91	1	1	1
4	junit4/src/main/java/org/junit/experimental/categories/Categories.java	219	249	junit4/src/main/java/org/junit/experimental/categories/Categories.java	351	366	0	1	0
5	junit4/src/main/java/org/junit/experimental/categories/Categories.java	219	249	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	1	0	0
6	junit4/src/main/java/org/junit/rules/TestWatchman.java	49	63	junit4/src/main/java/org/junit/experimental/theories/Theories.java	234	245	1	1	1
7	junit4/src/main/java/junit/runner/BaseTestRunner.java	155	174	junit4/src/main/java/junit/runner/BaseTestRunner.java	276	297	0	0	1
8	junit4/src/main/java/org/junit/runners/ParentRunner.java	406	423	junit4/src/main/java/org/junit/runners/ParentRunner.java	454	470	0	0	0
9	junit4/src/main/java/org/junit/runners/ParentRunner.java	406	423	junit4/src/main/java/org/junit/internal/runners/JUnit4ClassRunner.java	85	99	1	1	1
10	junit4/src/main/java/org/junit/runners/ParentRunner.java	406	423	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	1	1	1

0 means false or non-clone
1 means true or clone

11	junit4/src/main/java/org/junit/internal/management/ReflectiveRuntimeMXBean.java	42	58	junit4/src/main/java/org/junit/runners/ParentRunner.java	529	542	1	0	1
12	junit4/src/main/java/junit/textui/TestRunner.java	152	187	junit4/src/main/java/org/junit/runners/ParentRunner.java	477	510	0	0	0
13	junit4/src/main/java/org/junit/runners/Parameterized.java	455	470	junit4/src/main/java/org/junit/runners/ParentRunner.java	454	470	0	1	1
14	junit4/src/main/java/org/junit/runner/notification/SynchronizedRunListener.java	117	128	junit4/src/main/java/org/junit/runners/model/TestClass.java	315	328	1	1	1
15	junit4/src/main/java/org/junit/runner/notification/SynchronizedRunListener.java	117	128	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	37	48	1	1	0
16	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	60	82	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	106	126	0	1	1
17	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	60	82	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	57	84	1	1	1
18	junit4/src/main/java/junit/framework/ComparisonCompactor.java	21	32	junit4/src/main/java/org/junit/ComparisonFailure.java	95	106	1	1	1
19	junit4/src/main/java/junit/framework/ComparisonCompactor.java	21	32	junit4/src/main/java/org/junit/internal/ArrayComparisonFailure.java	50	65	0	0	0
20	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	56	71	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	73	88	1	1	1
21	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	56	71	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	1	1	1
22	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	56	71	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	22	37	1	1	1

0 means false or non-clone
1 means true or clone

23	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	56	71	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	39	54	1	1	1
24	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	56	71	junit4/src/main/java/org/junit/internal/builders/AnnotatedBuilder.java	101	115	0	0	0
25	junit4/src/main/java/org/junit/internal/runners/JUnit38ClassRunner.java	102	127	junit4/src/main/java/junit/framework/TestSuite.java	49	74	0	1	0
26	junit4/src/main/java/org/junit/internal/runners/JUnit38ClassRunner.java	102	127	junit4/src/main/java/org/junit/runners/parameterized/BlockJUnit4ClassRunnerWithParameters.java	59	94	0	0	0
27	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	123	143	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	0	0
28	junit4/src/main/java/junit/framework/TestSuite.java	49	74	junit4/src/main/java/junit/runner/BaseTestRunner.java	95	142	1	1	1
29	junit4/src/main/java/junit/framework/TestSuite.java	49	74	junit4/src/main/java/junit/framework/TestSuite.java	121	146	1	1	0
30	junit/runners/BlockJUnit4ClassRunner.java	91	105	junit4/src/main/java/org/junit/internal/runners/JUnit4ClassRunner.java	85	99	1	1	0
31	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	91	105	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	1	1	0
32	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	91	105	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	0	0	0
33	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	91	105	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	1	0	0

0 means false or non-clone
1 means true or clone

34	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	91	105	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	116	127	0	0	1
35	junit4/src/main/java/org/junit/runners/ParentRunner.java	325	338	junit4/src/main/java/junit/textui/TestRunner.java	112	123	0	0	0
36	junit4/src/main/java/org/junit/runners/ParentRunner.java	325	338	junit4/src/main/java/junit/runner/BaseTestRunner.java	253	264	0	0	1
37	junit4/src/main/java/org/junit/runners/ParentRunner.java	325	338	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	0	1	1
38	junit4/src/main/java/org/junit/rules/TemporaryFolder.java	232	251	junit4/src/main/java/org/junit/runner/manipulation/Orderer.java	28	48	0	0	0
39	junit4/src/main/java/org/junit/runner/FilterFactories.java	21	33	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	47	69	0	0	0
40	junit4/src/main/java/junit/extensions/ActiveTestSuite.java	39	54	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	1	1	1
41	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	119	146	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	170	188	0	1	0
42	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	119	146	junit4/src/main/java/org/junit/internal/runners/statements/ExpectException.java	15	37	1	1	0
43	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	119	146	junit4/src/main/java/org/junit/internal/runners/JUnit38ClassRunner.java	149	168	0	1	0
44	junit4/src/main/java/org/junit/experimental/theories/internal/Assignments.java	110	123	junit4/src/main/java/org/junit/experimental/categories/CategoryFilterFactory.java	36	50	1	1	1
45	junit4/src/main/java/org/junit/experimental/theories/internal/Assignments.java	110	123	junit4/src/main/java/org/junit/experimental/categories/Categories.java	351	366	1	1	1
46	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	0	0	0

0 means false or non-clone
1 means true or clone

47	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/runners/model/TestClass.java	244	258	0	0	1
48	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	0	0	0
49	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	144	158	1	1	1
50	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	0	0	1
51	junit4/src/main/java/org/junit/internal/runners/TestClass.java	72	85	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	0	0
52	junit4/src/main/java/org/junit/runners/ParentRunner.java	301	312	junit4/src/main/java/org/junit/experimental/theories/Theories.java	234	245	1	1	1
53	junit4/src/main/java/org/junit/runners/ParentRunner.java	301	312	junit4/src/main/java/org/junit/rules/ExpectedException.java	255	266	1	0	1
54	junit4/src/main/java/org/junit/experimental/theories/Theories.java	125	142	junit4/src/main/java/org/junit/experimental/theories/Theories.java	88	102	1	1	0
55	junit4/src/main/java/org/junit/experimental/theories/Theories.java	125	142	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	1	0
56	junit4/src/main/java/org/junit/experimental/theories/Theories.java	125	142	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	1	1	0
57	junit4/src/main/java/org/junit/rules/ErrorCollector.java	88	100	junit4/src/main/java/org/junit/rules/ErrorCollector.java	48	59	1	1	0
58	junit4/src/main/java/org/junit/ComparisonFailure.java	118	129	junit4/src/main/java/org/junit/ComparisonFailure.java	95	106	0	0	1
59	junit4/src/main/java/junit/textui/TestRunner.java	134	146	junit4/src/main/java/junit/textui/TestRunner.java	112	123	0	0	0
60	junit4/src/main/java/org/junit/internal/runners/ClassRoadie.java	51	67	junit4/src/main/java/junit/framework/TestCase.java	138	153	0	0	1
61	junit4/src/main/java/org/junit/internal/runners/ClassRoadie.java	51	67	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	128	144	1	1	1

0 means false or non-clone
1 means true or clone

62	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/junit/runner/BaseTestRunner.java	303	320	0	0	0
63	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/org/junit/rules/TestWatcher.java	96	108	0	0	0
64	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/org/junit/runner/notification/RunNotifier.java	170	182	0	0	0
65	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/org/junit/runners/ParentRunner.java	454	470	0	0	0
66	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	1	1	0
67	junit4/src/main/java/junit/textui/TestRunner.java	112	123	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	1	0	0
68	junit4/src/main/java/junit/framework/JUnit4TestAdapter.java	68	80	junit4/src/main/java/org/junit/rules/TestWatcher.java	96	108	0	1	0
69	junit4/src/main/java/junit/framework/JUnit4TestAdapter.java	68	80	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	88	99	0	0	0
70	junit4/src/main/java/junit/framework/JUnit4TestAdapter.java	68	80	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	116	127	1	1	0
71	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	247	264	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	170	188	0	1	0
72	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	247	264	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	214	237	0	1	1
73	junit4/src/main/java/org/junit/runner/manipulation/Ordering.java	88	99	junit4/src/main/java/org/junit/runner/manipulation/Ordering.java	31	45	0	0	0
74	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	junit4/src/main/java/org/junit/runners/model/TestClass.java	244	258	1	0	0
75	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	0	0	0

0 means false or non-clone
1 means true or clone

76	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	0	0	0
77	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	1	1	1
78	junit4/src/main/java/org/junit/runners/model/TestClass.java	63	74	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	1	0
79	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	49	62	junit4/src/main/java/org/junit/experimental/results/FailureList.java	16	27	0	0	0
80	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	49	62	junit4/src/main/java/org/junit/experimental/categories/Categories.java	204	217	1	1	1
81	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	49	62	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	116	127	0	0	0
82	junit4/src/main/java/org/junit/validator/AnnotationsValidator.java	60	76	junit4/src/main/java/org/junit/validator/AnnotationValidatorFactory.java	23	37	0	0	0
83	junit4/src/main/java/org/junit/validator/AnnotationsValidator.java	60	76	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	1	1	1
84	junit4/src/main/java/org/junit/validator/AnnotationsValidator.java	60	76	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	1	0
85	junit4/src/main/java/org/junit/validator/AnnotationsValidator.java	60	76	junit4/src/main/java/org/junit/experimental/categories/Categories.java	351	366	0	0	1
86	junit4/src/main/java/org/junit/validator/AnnotationsValidator.java	60	76	junit4/src/main/java/org/junit/internal/runners/rules/RuleMemberValidator.java	167	185	0	1	0
87	junit4/src/main/java/org/junit/runner/Computer.java	26	44	junit4/src/main/java/org/junit/internal/ComparisonCriteria.java	96	107	0	1	0
88	junit4/src/main/java/org/junit/runner/Computer.java	26	44	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	104	121	0	1	0

0 means false or non-clone
1 means true or clone

89	junit4/src/main/java/org/junit/ComparisonFailure.java	95	106	junit4/src/main/java/org/junit/Assert.java	110	122	0	0	0
90	junit4/src/main/java/org/junit/ComparisonFailure.java	95	106	junit4/src/main/java/org/junit/internal/ComparisonCriteria.java	96	107	1	0	0
91	junit4/src/main/java/org/junit/ComparisonFailure.java	95	106	junit4/src/main/java/org/junit/internal/ArrayComparisonFailure.java	50	65	0	0	0
92	junit4/src/main/java/org/junit/rules/TestWatcher.java	96	108	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	88	99	0	0	0
93	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	106	126	junit4/src/main/java/org/junit/internal/runners/statements/ExpectException.java	15	37	1	1	1
94	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	106	126	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	57	84	1	1	0
95	junit4/src/main/java/org/junit/runners/model/TestClass.java	244	258	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	0	0	0
96	junit4/src/main/java/org/junit/runners/model/TestClass.java	244	258	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	1	1	1
97	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	73	88	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	1	1	1
98	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	73	88	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	22	37	1	1	1
99	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	73	88	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	39	54	1	1	1
100	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	73	88	junit4/src/main/java/org/junit/internal/builders/AnnotatedBuilder.java	101	115	0	0	0

0 means false or non-clone
1 means true or clone

101	junit4/src/main/java/org/junit/internal/requests/FilterRequest.java	33	44	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	153	168	0	0	0
102	junit4/src/main/java/org/junit/internal/requests/FilterRequest.java	33	44	junit4/src/main/java/org/junit/experimental/results/FailureList.java	16	27	0	0	0
103	junit4/src/main/java/org/junit/internal/requests/FilterRequest.java	33	44	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	1	0	1
104	junit4/src/main/java/org/junit/runners/RuleContainer.java	71	85	junit4/src/main/java/org/junit/runners/RuleContainer.java	55	66	0	0	0
105	junit4/src/main/java/org/junit/internal/runners/SuiteMethod.java	27	40	junit4/src/main/java/junit/framework/TestSuite.java	284	297	1	0	0
106	junit4/src/main/java/org/junit/internal/runners/SuiteMethod.java	27	40	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	105	121	0	1	0
107	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	0	1	0
108	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	0	0
109	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	junit4/src/main/java/org/junit/experimental/theories/ParameterSignature.java	107	124	1	1	0
110	junit4/src/main/java/org/junit/runners/model/TestClass.java	82	97	junit4/src/main/java/junit/framework/TestCase.java	160	185	0	0	0
111	junit4/src/main/java/org/junit/experimental/theories/Theories.java	190	202	junit4/src/main/java/org/junit/experimental/theories/Theories.java	234	245	1	0	0
112	junit4/src/main/java/org/junit/runner/manipulation/Ordering.java	55	79	junit4/src/main/java/org/junit/runner/manipulation/Orderer.java	28	48	0	1	0
113	junit4/src/main/java/org/junit/rules/TestWatcher.java	53	76	junit4/src/main/java/org/junit/rules/TestWatcher.java	55	74	1	1	1
114	junit4/src/main/java/org/junit/rules/TestWatcher.java	53	76	junit4/src/main/java/org/junit/internal/runners/statements/ExpectException.java	15	37	1	1	0

0 means false or non-clone
1 means true or clone

115	junit4/src/main/java/org/junit/runner/notification/RunNotifier.java	170	182	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	0	0	0
116	junit4/src/main/java/org/junit/internal/builders/AnnotatedBuilder.java	80	91	junit4/src/main/java/org/junit/runners/parameterized/BlockJUnit4ClassRunnerWithParameters.java	190	202	1	0	1
117	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	170	188	junit4/src/main/java/org/junit/internal/runners/statements/FailOnTimeout.java	214	237	0	1	0
118	junit4/src/main/java/org/junit/experimental/theories/Theories.java	268	282	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	1	0	0
119	junit4/src/main/java/org/junit/experimental/max/MaxHistory.java	139	152	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	37	48	0	0	0
120	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	168	180	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	105	121	0	0	0
121	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	168	180	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	70	82	0	0	0
122	junit4/src/main/java/org/junit/runners/model/NoGenericTypeParametersValidator.java	24	35	junit4/src/main/java/org/junit/runners/parameterized/TestWithParameters.java	55	70	1	0	0
123	junit4/src/main/java/org/junit/rules/TestWatcher.java	55	74	junit4/src/main/java/org/junit/internal/runners/statements/ExpectException.java	15	37	1	1	1
124	junit4/src/main/java/org/junit/rules/TestWatcher.java	55	74	junit4/src/main/java/org/junit/rules/ExternalResource.java	46	67	1	1	1
125	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	146	157	junit4/src/main/java/org/junit/internal/runners/ClassRoadie.java	69	80	1	1	1
126	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	146	157	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	39	55	0	0	0

0 means false or non-clone
1 means true or clone

127	junit4/src/main/java/org/junit/runners/ParentRunner.java	454	470	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	0	0	0
128	junit4/src/main/java/org/junit/experimental/theories/Theories.java	234	245	junit4/src/main/java/org/junit/rules/ExpectedException.java	255	266	1	0	0
129	junit4/src/main/java/org/junit/internal/TextListener.java	59	73	junit4/src/main/java/org/junit/internal/TextListener.java	80	92	1	1	0
130	junit4/src/main/java/org/junit/internal/ComparisonCriteria.java	96	107	junit4/src/main/java/org/junit/internal/ArrayComparisonFailure.java	50	65	1	1	0
131	junit4/src/main/java/org/junit/validator/AnnotationValidatorFactory.java	23	37	junit4/src/main/java/org/junit/experimental/categories/CategoryFilterFactory.java	36	50	0	0	0
132	junit4/src/main/java/org/junit/validator/AnnotationValidatorFactory.java	23	37	junit4/src/main/java/org/junit/experimental/categories/Categories.java	351	366	0	0	0
133	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	99	110	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	144	158	0	0	0
134	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	99	110	junit4/src/main/java/org/junit/experimental/theories/Theories.java	144	157	0	1	0
135	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	99	110	junit4/src/main/java/org/junit/runners/Parameterized.java	321	335	1	0	0
136	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	99	110	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	1	1	1
137	junit4/src/main/java/org/junit/internal/runners/statements/ExpectException.java	15	37	junit4/src/main/java/org/junit/internal/management/ReflectiveThreadMXBean.java	48	69	0	1	1
138	junit4/src/main/java/org/junit/runners/model/FrameworkMethod.java	144	158	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	0	0	0

0 means false or non-clone
1 means true or clone

139	junit4/src/main/java/org/junit/runners/model/MultipleFailureException.java	86	103	junit4/src/main/java/org/junit/rules/ErrorCollector.java	48	59	0	0	1
140	junit4/src/main/java/org/junit/runners/model/MultipleFailureException.java	86	103	junit4/src/main/java/org/junit/experimental/results/FailureList.java	16	27	0	0	1
141	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	junit4/src/main/java/org/junit/runners/Parameterized.java	321	335	1	0	0
142	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	junit4/src/main/java/junit/framework/TestSuite.java	284	297	0	0	0
143	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	1	1	0
144	junit4/src/main/java/org/junit/internal/runners/TestClass.java	42	57	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	0	1	0
145	junit4/src/main/java/org/junit/experimental/categories/CategoryValidator.java	43	55	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	0	1	0
146	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	39	55	junit4/src/main/java/org/junit/experimental/results/FailureList.java	16	27	0	0	0
147	junit4/src/main/java/org/junit/internal/runners/MethodRoadie.java	39	55	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	0	0	1
148	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	22	37	junit4/src/main/java/org/junit/experimental/theories/internal/SpecificDataPointsSupplier.java	39	54	1	1	1
149	junit4/src/main/java/org/junit/experimental/theories/Theories.java	144	157	junit4/src/main/java/org/junit/experimental/theories/Theories.java	88	102	0	1	1
150	junit4/src/main/java/org/junit/experimental/theories/Theories.java	144	157	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	0	1	1
151	junit4/src/main/java/org/junit/internal/management/ManagementFactory.java	24	41	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	1	1	0

0 means false or non-clone
1 means true or clone

152	junit4/src/main/java/org/junit/rules/ExternalResource.java	46	67	junit4/src/main/java/org/junit/rules/ExternalResource.java	48	65	1	1	1
153	junit4/src/main/java/org/junit/rules/ExternalResource.java	46	67	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	1	1	0
154	junit4/src/main/java/org/junit/rules/ExternalResource.java	46	67	junit4/src/main/java/org/junit/internal/runners/statements/RunAfters.java	23	40	1	1	1
155	junit4/src/main/java/org/junit/runners/Parameterized.java	321	335	junit4/src/main/java/org/junit/experimental/theories/Theories.java	88	102	1	1	1
156	junit4/src/main/java/org/junit/runners/Parameterized.java	321	335	junit4/src/main/java/org/junit/experimental/theories/ParameterSignature.java	107	124	1	1	0
157	junit4/src/main/java/org/junit/runners/Parameterized.java	321	335	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	0	1	0
158	junit4/src/main/java/org/junit/runners/ParentRunner.java	388	404	junit4/src/main/java/org/junit/experimental/theories/ParameterSignature.java	107	124	0	0	0
159	junit4/src/main/java/org/junit/runners/parameterized/BlockJUnit4ClassRunnerWithParameters.java	114	144	junit4/src/main/java/org/junit/runners/parameterized/BlockJUnit4ClassRunnerWithParameters.java	59	94	1	1	0
160	junit4/src/main/java/org/junit/runners/ParentRunner.java	429	452	junit4/src/main/java/org/junit/internal/runners/JUnit38ClassRunner.java	149	168	1	0	0
161	junit4/src/main/java/org/junit/experimental/theories/Theories.java	88	102	junit4/src/main/java/org/junit/experimental/theories/Theories.java	104	118	1	1	1
162	junit4/src/main/java/org/junit/rules/ExternalResource.java	48	65	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	0	0	0
163	junit4/src/main/java/org/junit/rules/ExternalResource.java	48	65	junit4/src/main/java/org/junit/internal/runners/statements/RunAfters.java	23	40	1	1	1

0 means false or non-clone
1 means true or clone

164	junit4/src/main/java/org/junit/internal/runners/JUnit4ClassRunner.java	85	99	junit4/src/main/java/junit/framework/JUnit4TestAdapterCache.java	44	63	0	0	0
165	junit4/src/main/java/org/junit/internal/runners/JUnit4ClassRunner.java	85	99	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	1	1	1
166	junit4/src/main/java/org/junit/rules/TestWatchman.java	46	65	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	1	1	1
167	junit4/src/main/java/org/junit/runners/Parameterized.java	442	453	junit4/src/main/java/org/junit/runners/Parameterized.java	397	408	0	0	0
168	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	88	99	junit4/src/main/java/org/junit/internal/AssumptionViolatedException.java	91	110	0	1	0
169	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	88	99	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	37	48	1	1	1
170	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	88	99	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	70	82	1	1	1
171	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	0	0	1
172	junit4/src/main/java/org/junit/experimental/theories/Theories.java	230	248	junit4/src/main/java/org/junit/internal/runners/statements/RunAfters.java	23	40	1	0	0
173	junit4/src/main/java/org/junit/experimental/results/FailureList.java	16	27	junit4/src/main/java/org/junit/internal/requests/MemoizingRequest.java	13	26	1	0	1
174	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	junit4/src/main/java/org/junit/experimental/theories/ParameterSignature.java	107	124	1	0	0
175	junit4/src/main/java/org/junit/runners/model/TestClass.java	278	300	junit4/src/main/java/junit/framework/TestCase.java	160	185	0	0	0

0 means false or non-clone
1 means true or clone

176	junit4/src/main/java/org/junit/experimental/theories/PotentialAssignment.java	29	45	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	116	127	1	0	0
177	junit4/src/main/java/org/junit/runner/JUnitCore.java	131	143	junit4/src/main/java/org/junit/runners/ParentRunner.java	361	374	0	1	1
178	junit4/src/main/java/org/junit/experimental/theories/ParameterSignature.java	107	124	junit4/src/main/java/org/junit/internal/runners/rules/RuleMemberValidator.java	167	185	0	0	1
179	junit4/src/main/java/org/junit/runners/model/TestClass.java	315	328	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	37	48	0	1	0
180	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	47	69	junit4/src/main/java/org/junit/runner/manipulation/Filter.java	104	121	1	1	1
181	junit4/src/main/java/org/junit/experimental/categories/Categories.java	351	366	junit4/src/main/java/org/junit/experimental/max/MaxCore.java	105	121	0	0	0
182	junit4/src/main/java/junit/runner/BaseTestRunner.java	253	264	junit4/src/main/java/org/junit/runners/ParentRunner.java	529	542	0	0	1
183	junit4/src/main/java/junit/runner/BaseTestRunner.java	253	264	junit4/src/main/java/org/junit/runners/BlockJUnit4ClassRunner.java	116	127	0	1	0
184	junit4/src/main/java/org/junit/internal/Throwables.java	85	106	junit4/src/main/java/org/junit/internal/Throwables.java	130	154	1	1	1
185	junit4/src/main/java/org/junit/internal/TextListener.java	80	92	junit4/src/main/java/junit/textui/ResultPrinter.java	78	92	1	1	1
186	junit4/src/main/java/org/junit/runners/Parameterized.java	397	408	junit4/src/main/java/org/junit/runners/ParentRunner.java	529	542	0	0	0
187	junit4/src/main/java/org/junit/internal/AssumptionViolatedException.java	91	110	junit4/src/main/java/org/junit/experimental/results/ResultMatchers.java	70	82	0	0	0

0 means false or non-clone
1 means true or clone

Natty Manual Validation Result

Total = 9 clone pairs True Positive = 7 pairs False Positive = 2 pairs

Clone Pair Number	File Path 1	Start line 1	End line 1	File Path 2	Start line 2	End line 2	Judge 1 (Vara)	Judge 2 (Wannaporn)	Judge 3 (Weekit)
1	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	207	236	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	334	380	0	1	1
2	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	207	236	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	551	581	1	0	1
3	nattyw-otest/main/java/com/joestelmach/natty/Parser.java	186	250	nattyw-otest/main/java/com/joestelmach/natty/Parser.java	260	326	1	1	0
4	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	289	314	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	169	194	1	0	1
5	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	289	314	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	243	271	1	1	1
6	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	169	194	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	243	271	1	1	1
7	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	169	194	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	80	108	1	1	1
8	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	334	380	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	551	581	1	0	0
9	nattyw-otest/main/java/com/joestelmach/natty/WalkerState.java	446	524	nattyw-otest/main/java/com/joestelmach/natty/Parser.java	79	159	0	0	0

0 means false or non-clone
1 means true or clone

Merry Manual Validation Result

Total = 8 clone pairs True Positive = 6 pairs False Positive = 2 pairs

Clone Pair Number	File Path 1	Start line 1	End line 1	File Path 2	Start line 2	End line 2	Judge 1 (Vara)	Judge 2 (Wannaporn)	Judge 3 (Weekit)
1	MerryEngine/JavaExtractor/JPredict/src/main/java/JavaExtractor/Visitors/FunctionVisitor.java	25	40	MerryEngine/JavaExtractor/JPredict/src/main/java/JavaExtractor/Visitors/FunctionVisitor.java	42	55	0	0	0
2	MerryEngine/src/main/java/MethodPair.java	87	105	MerryEngine/src/main/java/MethodPair.java	72	85	0	1	1
3	MerryEngine/src/main/java/MethodPair.java	87	105	MerryEngine/src/main/java/MethodPair.java	107	121	1	0	1
4	MerryEngine/src/main/java/JavaMethodParser.java	34	58	MerryEngine/src/main/java/JavaMethodParser.java	61	84	0	1	1
5	MerryEngine/src/main/java/JavaMethodParser.java	34	58	MerryEngine/src/main/java/JavaMethodParser.java	88	112	0	1	0
6	MerryEngine/JavaExtractor/JPredict/src/main/java/JavaExtractor/ExtractFeaturesTask.java	36	52	MerryEngine/JavaExtractor/JPredict/src/main/java/JavaExtractor/ExtractFeaturesTask.java	54	67	1	0	1
7	MerryEngine/src/main/java/JavaMethodParser.java	61	84	MerryEngine/src/main/java/JavaMethodParser.java	88	112	1	1	1
8	MerryEngine/src/main/java/MethodPair.java	72	85	MerryEngine/src/main/java/MethodPair.java	107	121	1	1	1

0 means false or non-clone
1 means true or clone