ABERYSTWYTH UNIVERSITY

PROJECT REPORT FOR CS39440 MAJOR PROJECT

IntelliJ Plugin to Aid With Plagiarism Detection

Author:
Darren S. WHITE
(daw48@aber.ac.uk)

Supervisor:
Chris LOFTUS (cwl@aber.ac.uk)

April 9, 2018

Version: 1.0 (draft)

This report was submitted as partial fulfilment of a BSc degree in

Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, United Kingdom

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract

Source code plagiarism is an ever-growing issue in academia, primarily in Computer Science. The majority of tools that exist to detect plagiarism only analyse the final piece of code. This paper shows the research and development of a new tool which will detect how the code was written. This tool will be an IntelliJ IDEA plugin and will track file changes in the editor. The tracked data will give more of an insight into how plagiarism evolves over the course of development. This method of detection would allow direct identification of the specific pieces of code that were plagiarised.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Overview

Plagiarism is becoming popular among students due to ease of access to the internet. Plagiarism.org states that, "One out of three high school students admitted that they used the Internet to plagiarize an assignment" [1]. Detecting acts of plagiarism is not a simple task, especially when done manually. Humans are simply not capable of analysing multiple pieces of work and finding similarities. At least not efficiently or quickly. Automatic detection systems already exist but they only analyse the final piece of work (these systems are described more in detail in section 2.1). These systems can be improved by advancing the detection algorithms. This may prove difficult over time as these algorithms become more complex.

A system that analyses work from its inception would allow more targeted and sophisticated detection methods to be performed. These methods could directly identify the plagiarised work, and how it may be transformed to hide any evidence. This system would not act as a sole detection system, but instead alongside existing tools. This would improve the detection rate and provide more evidence for such cases.

1.2 Plagiarism and Unacceptable Academic Practice

Aberystwyth University classifies plagiarism, collusion, and fabrication of evidence of data as acts of UAC (Unacceptable Academic Practice) [2]. The act of plagiarising is to commit fraud by stealing someone's work and not clearly referencing the owner [3]. Plagiarism has many forms, some of these are described below [2] [3] [4] [5].

- Copying or cloning another's work without modification (including copying and pasting)
- Paraphrasing or modifying another's work without due acknowledgement
- Using a quotation with incorrect information about the source
- The majority of the work is made up from other sources
- Copying an original idea from another persons work

- Putting your own name on someone else's work

These forms of plagiarism apply to written work as well as source code. In terms of coding, copying someone's code without modification or acknowledgement is still plagiarism. Automated systems can be put in place to detect plagiarism and UAC.

The main objective of detecting plagiarism, is finding pieces of work which originated from another source. The plagiarised work can often be obfuscated or modified in a way to try and conceal the original work yet keep the same outcome. In source code, this ranges from the simple changes to more complex alterations. The program plagiarism spectrum describes the levels of plagiarism that can be done, with level 0 having no modifications and level 6 altering the control flow of the program [6]. The higher levels make it more difficult to compare pieces of work against each other.

Chapter 2

Background

This project introduces three problems. Tracking the work as it is being written, and identifying when plagiarism has occurred. Tracking code being written will be accomplished by developing a plugin for IntelliJ IDEA. IntelliJ IDEA is a Java IDE (Integrated Development Environment) for software development. IntelliJ has to capability to add and develop plugins. These plugins are developed using the IntelliJ Platform [7]. Identifying plagiarism is the second major task for this project. Due to this system operating differently to existing systems, it will be difficult to determine how to accurately detect plagiarism.

"Plagiarism detection usually is based on comparison of two or more documents" [8]. This is what makes this project stand out for me. It's not simply reinventing the wheel, but finding new ways to solve problems which still exist in academia.

2.1 Existing Systems

The most popular existing systems are Turnitin, MOSS, JPlag, and YAP3, amongst many others. For this system, MOSS and JPlag are worth looking into as they both check source code, whereas Turnitin only checks plain text [8]. Although these tools can automatically identify plagiarism, they still require human verification.

MOSS relies on the Winnowing detection algorithm. It works by creating fingerprints or hashes for documents [9]. Creating single hashes for each document allows for exact document comparisons. Single hashes are useful for checking if a document is correct and non-corrupt. Instead, Winnowing, uses multiple k-grams for each document. K-grams all for partial document comparisons between multiple documents. Comparing between multiple sources does however require a large set of documents beforehand.

JPlag provides an online user interface where documents can be submitted and the results can be viewed. JPlag parses each document into token strings and these tokens are compared in pairs between documents. The percentage of tokens that match is referred to as the similarity factor [10].

Yap3 operates in a similar way to JPlag and MOSS. It uses its own similarity detection algorithm, RKR-GST. The algorithm compares sets of strings in a text much like the other algorithms [11].

Plagiarism detection tools can use either extrinsic or intrinsic detection algorithms. Extrinsic detection uses external sources to compare against. Using a massive collection of external documents, extrinsic algorithms can be used very effectively although will take a large amount of time to process. Intrinsic detection analyses the document to detect changes in writing style. This allows the post-processing time to be very small in comparison to extrinsic algorithms. All of the existing systems described above use extrinsic detection algorithms.

2.2 IntelliJ Plugin SDK

Being unfamiliar with the IntelliJ Plugin SDK, I delved deep into the online tutorials provided by JetBrains [12]. IntelliJ comes bundled with the IntelliJ Platform Plugin SDK so setting up the development environment is no issue. The IntelliJ Community Edition is open source and contains many plugins [13]. This repository is very useful. Looking at existing plugins is sometimes more useful than reading tutorials.

One aspect of the plugin that would be needed is to track keyboard events. I set out to try and find what possible methods there were of doing this. This feature is not mentioned in the tutorial, so I went digging through the SDK in the Community Edition repository. TypedHandlerDelegate and TypedHandler are classes used to perform actions upon typing events in the editor of the IDE.

Saving data to disk is also another feature that would be used by the plugin. This feature is used widely by many plugins and was documented in the tutorial. The tutorial was understandable but I decided to take a look at an existing plugin for a real example, the GitHub plugin. GitHub saves settings to file and so this was helpful to my understanding.

2.3 Back-end Server

The following is a comprehensive list of the possible technologies that could be used for the back-end server.

- **Python / Flask** - Micro web framework for Python based on Werkzeug, and Jinja2.
- **Ruby on Rails** - Server-side web framework written in Ruby which uses a MVC architecture and provides default structures. It is very quick to implement a solution.
- **Bootstrap / JQuery** - Bootstrap is a front-end library for HTML, CSS, and JavaScript. JQuery is a JavaScript library.

2.4 Analysis

2.4.1 Specification

2.4.2 Potential Issues

2.5 Process

The approach I choose to use for this project was an agile methodology, scrum. Scrum uses short iterations, each consisting of planning at the beginning, implementation, review, and then retrospective at the end. Weekly meetings on Mondays with my supervisor were also organised. These meetings consisted mostly of discussions of the previous and next sprints.

Planning involves discussing and deciding which stories should be worked on during the sprint. A story is a piece of work that needs to be done. The intricate details of each story may not yet be known but they will develop over the course of the iteration. A story will have a time estimate associated with it. The golden ratio is used as a guideline, and the story points are described below.

- **1** - 10 minutes to 1 hour
- **2** - A few hours to half a day
- **3** - A few days
- **5** - A week
- **8** - Over a week, this story should be broken into smaller stories

Implementation and review take up most of the iteration time. This is spent designing, developing, and reviewing code that will end up in the code base. Once code has been reviewed for a story, it can be marked as done.

Retrospective is a reflective process. It is a discussion of what went well, what didn't go well, and what could change for the next sprint. The retrospective is aimed to improve the scrum process over time.

To track each sprint and its stories, I used milestones and issues on GitHub [14]. During planning I would create a new milestone, assign issues to it (creating new issues if necessary), and set a goal. The goal would be a general aim for that sprint, which multiple stories would accomplish. During implementation and review, issues can easily be closed by referencing them in a commit message with specific keywords such as *Fixes #IssueNum* [15]. After the sprint is done, I would close the milestone. Any remaining issues in the milestone would remain in the backlog still marked as open.

Chapter 3

Design Architecture

3.1 Final Design

Chapter 4

Iteration 0

4.1 Planning

4.2 Implementation

4.3 Retrospective

Chapter 5

Iteration 1

5.1 Planning

5.2 Implementation

5.3 Retrospective

Chapter 6

Iteration 2

6.1 Planning

6.2 Implementation

6.3 Retrospective

Chapter 7

Iteration 3

7.1 Planning

7.2 Implementation

7.3 Retrospective

Chapter 8

Iteration 4

8.1 Planning

8.2 Implementation

8.3 Retrospective

Chapter 9

Iteration 5

9.1 Planning

9.2 Implementation

9.3 Retrospective

Chapter 10

Iteration 6

10.1 Planning

10.2 Implementation

10.3 Retrospective

Chapter 11

Iteration 7

11.1 Planning

11.2 Implementation

11.3 Retrospective

Chapter 12

Iteration 8

12.1 Planning

12.2 Implementation

12.3 Retrospective

Chapter 13

Testing

13.1 Plugin

13.2 Server

13.3 Post-Processor

Chapter 14

Evaluation

Appendices

Appendix A

Third-Party Code and Libraries

Appendix B

Code Examples

Annotated Bibliography

[1] Plagiarism.org. (2018) Plagiarism: Facts & stats - plagiarism.org. [Online]. Available: http://plagiarism.org/article/plagiarism-facts-and-stats

This web page contains results from surveys performed in schools on plagiarism. The outcomes from these surveys show how many students commit plagiarism.

[2] A. University. (2018) Aberystwyth university - regulation on unacceptable academic practice. [Online]. Available: https://www.aber.ac.uk/en/aqro/handbook/regulations/uap/

This shows Aberystwyth University's regulation on UAC and plagiarism. It describes the definition of UAC and the multiple forms it can take.

[3] Plagiarism.org. (2018) What is plagiarism? - plagiarism.org. [Online]. Available: http://plagiarism.org/article/what-is-plagiarism

This web page has detailed information on what exactly plagiarism is. It states what the definition of plagiarism is and example forms of plagiarism.

[4] Turnitin. (2018) Turnitin - the plagiarism spectrum. [Online]. Available: http://turnitin.com/assets/en_us/media/plagiarism-spectrum/

This web page shows a list of 10 different ways to plagiarise with example texts.

[5] P. Clough and D. O. I. Studies, "Old and new challenges in automatic plagiarism detection," in *National Plagiarism Advisory Service, 2003; http://ir.shef.ac.uk/cloughie/index.html*, 2003, pp. 391–407.

In-depth details on lexical and structural changes in program code plagiarism.

[6] A. Parker and J. Hamblen, "Computer algorithms for plagiarism detection," *IEEE Transactions on Education*, vol. 32, no. 2, pp. 94–99, may 1989. [Online]. Available: https://doi.org/10.1109/13.28038

Details on the plagiarism spectrum. The different levels of plagiarism. Level 0 has no changes and level 6 has the control logic changed.

[7] JetBrains. (2018) What is the intellij platform? [Online]. Available: http://www.jetbrains.org/intellij/sdk/docs/intro/intellij_platform.html

This page describes what the IntelliJ Platform is.

[8] R. Lukashenko, V. Graudina, and J. Grundspenkis, “Computer-based plagiarism detection methods and tools,” in *Proceedings of the 2007 international conference on Computer systems and technologies - CompSysTech '07*. ACM Press, 2007. [Online]. Available: https://doi.org/10.1145/1330598.1330642

Useful attribute table containing information on tools for detecting plagiarism such as Turnitin, and MOSS.

[9] S. Schleimer, D. S. Wilkerson, and A. Aiken, “Winnowing: local algorithms for document fingerprinting,” in *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD '03*. ACM Press, 2003. [Online]. Available: https://doi.org/10.1145/872757.872770

This went into detail on the ideas behind MOSS, such as document fingerprints and k-grams.

[10] L. Prechelt and G. Malpohl, “Finding plagiarisms among a set of programs with jplag,” vol. 8, 03 2003.

Detailed information on how JPlag detects plagiarism using tokens.

[11] M. Wise, “Yap3: Improved detection of similarities in computer program and other texts,” vol. 28, 04 1996.

In-depth detail on the RKR-GST algorithm used by YAP3 and the comparison between the YAP versions.

[12] JetBrains. (2018) Creating your first plugin. [Online]. Available: https://www.jetbrains.org/intellij/sdk/docs/basics/getting_started.html

The JetBrains tutorial for developing an IntelliJ Plugin.

[13] ——. (2018) Jetbrains/intellij-community: Intellij idea community edition. [Online]. Available: https://github.com/jetBrains/intellij-community

The IntelliJ IDEA Community Edition GitHub repository provides code for plugins. This helped with understanding the IntelliJ Platform Plugin SDK.

[14] GitHub. (2018) About milestones - user documentation. [Online]. Available: https://help.github.com/articles/about-milestones/

Describes using issues and milestones on GitHub.

[15] ——. (2018) Closing issues using keywords - user documentation. [Online]. Available: https://help.github.com/articles/closing-issues-using-keywords/

Explains how to close issues using commit messages by including specific keywords.

[16] U. Bandara and G. Wijayrathna, “Detection of source code plagiarism using machine learning approach,” *International Journal of Computer Theory and Engineering*, pp. 674–678, 2012. [Online]. Available: https://doi.org/10.7763/ijcte.2012.v4.555

[17] P. S, R. R, and S. B. B, “A survey on plagiarism detection,” *International Journal of Computer Applications*, vol. 86, no. 19, pp. 21–23, jan 2014. [Online]. Available: https://doi.org/10.5120/15104-3428

[18] Flask. (2018) Uploading files - flask documentation (0.12). [Online]. Available: http://flask.pocoo.org/docs/0.12/patterns/fileuploads/

[19] A. J. J. Davis. (2013) Test mongodb failure scenarios with mockupdb. [Online]. Available: https://emptysqua.re/blog/test-mongodb-failures-mockupdb/