Style Analysis for Source Code Plagiarism Detection – an Analysis of a Dataset of Student Coursework

Olfat M. Mirza and Mike Joy
Department of Computer Science
University of Warwick
Coventry, CV4 7AL, UK
{O.M.Mirza,M.S.Joy}@warwick.ac.uk

Georgina Cosma
School of Science and Technology
Nottingham Trent University
Nottingham, NG1 4FQ, UK
Georgina.Cosma@ntu.ac.uk

Abstract—**Plagiarism has become an increasing problem in higher education in recent years. Coding style can be used to detect source code plagiarism that involves writing and deciding the structure of the code which does not affect the logic of a program, thus offering a way to differentiate between different code authors. This paper focuses to identify whether a data set consisting of student programming assignments is rich enough to apply coding style metrics to detect similarities between code sequences, and we use the BlackBox dataset as a case study.**

Keywords-component; Source Code Plagairism Detection; Style Analysis; Coding Style

I. INTRODUCTION

The term *plagiarism* refers to reusing, paraphrasing, or copying work of somebody without giving them any credits and recognition, and includes attempting to showcase plagiarised content as one's own work. Hannabuss [2] defined plagiarism as "the unauthorized use or close imitation of the ideas and language/expression of someone else".

Source code plagiarism has a very significant definition and Parker and Hamblen [3] defined software plagiarism as "A program that has been produced from another program with a small number of routine transformations".

Detection of source code plagiarism has been analysed in various contexts [1], but research on the analysis of coding style for large datasets is limited. We investigate a dataset consisting of genuine student programming assignment submissions to determine if it is sufficiently rich to form a basis for coding style analysis, and in order to achieve this, we used the BlackBox source code dataset [5]. In this paper, content analysis is used to find out how suitable random samples taken from the dataset are.

II. BACKGROUND AND RELATED WORK

Significant research has been done on how to identify source code plagiarism [6]. One approach to source code plagiarism detection attempts to identify the authorship of the code from the way the code is written, the "coding style", which may be derived from coding conventions (sets of guidelines for a programming language, perhaps defined for use by a particular institution or company). These conventions usually cover such aspects as file organization, indentation, comments, declarations, use of white space, naming of variables, programming practices, programming principles, programming rules of thumb, and architectural best practices.

Coding style is a characteristic which can be used to detect source code plagiarism because it relates to programmer personality but does not affect the logic of a program and can thus be used to differentiate between different code fragments which are functionally similar [7].

III. PROPOSED METHOD

In this paper, we perform a content analysis on random samples of source code files taken from a large data set of student coursework submissions in order to identify whether the dataset contains sufficient files on which such a technique is likely to work.

A. BlackBox Dataset

BlackBox is a project that collects data from users of the BlueJ online educational software tool. BlueJ is a Java integrated development environment (IDE) designed for beginners, and BlueJ is free and open source software [4]. BlackBox contains code written by a wide variety of programmers, ranging from complete beginners to professional software developers.

B. Source Code Dataset and Sample Size

In this exploratory study, one of the focal issues was to determine the intended sample size suitable to be used in this and in future studies. This cannot be determined by number only, and common factors include the aim of the study, the population size and the sampling error. This paper considers random samples downloaded from BlackBox, and containing 250 Java files each. The sample datasets were downloaded from BlackBox to justify the study of coding style analysis.

C. Preprocessing the Source Code Files

Source code file preprocessing was applied in this study and although the file name and the real ID of the author are hidden in the code using hashes, each file has its own author. The task of preprocessing the files was performed using the following metrics:

1) *Removing any white space;*
2) *Removing the file header.*

IV. Experiment

In order to perform the experiment, we designed a small program based on the Java programming language which initially performs a random sample fetch from the BlackBox dataset. BlackBox contains some duplicated files (identified by having the same ID), and thus the fetcher was designed to choose one file if there is more than one file with the same ID to avoid duplication. The second stage is to count the number of lines and the size of each source code file. This is followed by measuring the complexity of each of the files by (for example) counting the number of loops and finding common loop words such as: for, if, if-else and while. When the system identified the features, the next stage was to group them according to pre-defined categories (see below).

A. Grouping the Source Code Files

It is considered important to have a case study of what types of source code files BlackBox contains. The random groups of source code files were downloaded from the BlackBox dataset and thereafter subgroups were created based on features including the number of lines per source code file and the complexity of the code in each file.

Five main subgroups, based on the number of code lines and the complexity of code, were identified for the each random group.

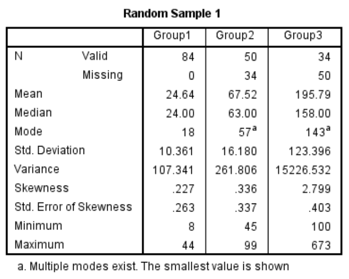
1. The first subgroup contains files which are “templates” or “common ground files”.
2. The second subgroup consists of short and simple code files. The length of the code is less than 40 lines and the maximum level of loops is 2.
3. The third subgroup consists of simple code files. The length of the code is on average more than 40 lines and less than 100 lines. The level of loops is more than 3 and includes some nested loops.
4. The fourth supgroup consists of code files which are long and complex.
5. The fifth supgroup contains files which are incomplete or empty.

B. Experimental Results

The main objective of the statistical analysis is to validate the grouping methods and to find out how rich the dataset can be in order to identify the coding style for the purpose of detecting plagiarism in source code. Since the first and the fifth groups contain files which a detection algorithm can safely ignore, they have not been used in the analysis discussed in this section. The statistics of first random sample are presented in Table 1.

In this analysis of the BlackBox source code there were some clear indications that some portions of the BlackBox source code would be usable for this coding style analysis study. It is clear that group three got the highest number of lines, and it gives a rich style analysis of this group. This suggests that the random samples are representative of the files contained in the BlackBox source code dataset. According to the feature analysis based on physical attributes that were applied to extract coding style, the number of lines was one of the main features considered when categorising the random sample into five subgroups. The analysis of the results shows that the three subgroups (subgroups 2-4) offer a rich source to which coding style analysis can be applied for the purpose of detecting plagiarism.

Table 1



Random Sample 1

		Group1	Group2	Group3
N	Valid	84	50	34
	Missing	0	34	50
Mean		24.64	67.52	195.79
Median		24.00	63.00	158.00
Mode		18	57[a]	143[a]
Std. Deviation		10.361	16.180	123.396
Variance		107.341	261.806	15226.532
Skewness		.227	.336	2.799
Std. Error of Skewness		.263	.337	.403
Minimum		8	45	100
Maximum		44	99	673

a. Multiple modes exist. The smallest value is shown

V. Conclusion and Future Work

This paper explores the suitability of methods based on coding style analysis which unite a content based analysis with random samples. The results suggest that the BlackBox source code dataset of student coursework is suitable for applying coding style based plagiarism detection techniques, since such a dataset contains sufficient files which are rich enough for such an analysis to be meaningful.

VI. References

[1] O. Mirza, M. Joy, "Style analysis for source code plagiarism detection." *Plagiarism Across Europe and Beyond 2015: Conference Proceedings.* pp. 53-61, 2015.

[2] S. Hannabuss, "Contested texts: issues of plagiarism," *Library management,* vol. 22, no. 6/7, pp. 311-318, 2001.

[3] A. Parker and J. Hamblen, "Computer algorithms for plagiarism detection," *IEEE Transactions on Education,* vol. 32, no. 2, pp. 94-99, 1989.

[4] M. Kolling, "Lessons from the Design of Three Educational Programming Environments: Blue, BlueJ and Greenfoot," *International Journal of People-Oriented Programming (IJPOP),* vol. 4, no. 1, pp. 5-32, 2015.

[5] N. Brown, M. Kolling, D. McCall and I. Utting, "Blackbox: A large scale repository of novice programmers' activity," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014

[6] M. Joy and G. Cosma, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis," *IEEE Transactions on Computers,* vol. 61, no. 3, pp. 379-394, 2012.

[7] S. Burrows, A. Uitdenbogerd and A. Turpin, "Application of information retrieval techniques for source code authorship attribution," *nternational Conference on Database Systems for Advanced Applications,* pp. 699-713, 2009.