Liu, C.-C. et al. (Eds.) (2014). Proceedings of the 22nd International Conference on Computers in Education. Japan: Asia-Pacific Society for Computers in Education

An Online System for Scoring and Plagiarism Detection in University Programing Class

Asako OHNO*, Takahiro YAMASAKI, and Kin-ichioroh TOKIWA
Faculty of Engineering, Osaka Sangyo University, Japan
*ohno@eic.osaka-sandai.ac.jp

Abstract: Scoring and plagiarism detection in university programming classes are important but time-consuming and burdensome tasks for teachers. In this paper, we explain about the structure and functions of the online education support system developed for university programming class. The system mainly provides two functions: scoring function and plagiarism detection function implementing two different kind of similarity measuring methods that had been proposed in the previous studies. Each of the methods calculates similarity between a pair of source codes in different aspect: content-based similarity and style-based similarity. This paper also describes how those two different methods work for the system to provide each functions with summarized explanations of each of the methods.
Keywords: Source code similarity, programming education, online education support system

1. Introduction

There have been many education support systems proposed for programming class. Along with the recent diffusion of Java in university programming classes, many of the system are aiming to score or detect plagiarism among Java source files (Ihantola, et al., 2010). Another survey said it was difficult to evaluate which one of the proposed systems was the most efficient because many of them applied the system to the source codes with were in designated formats (Queirós and Leal, 2012).

There are some issues to be considered regarding the nature of source codes produced as assignments in programming classes when providing scoring and plagiarism detection function. For example, source codes produced as assignments are generally short in length so that it difficult to quantify feature that contains rich enough information to represent its algorithmic or structural features. This characteristic makes it difficult to measure similarity between model answer and students' source codes. It may also difficult to find plagiarism from a pair of short source codes. Further, regarding plagiarism detection function, it is difficult to distinguish similarities in nature with the ones caused by plagiarism because generally, source codes produced in programming classes are produced to achieve the same purpose and the students are often ordered to use the same algorithms that they had just learned in their class. We have developed an education support system for (mainly Java, but also applicable for C) programming class that implemented two similarity measuring methods: FRef (Ohno and Murao, 2007) that quantifies content-based similarity from short source codes by using arbitrary-chosen reference source codes as scoring function and CM Algorithm (Ohno and Murao, 2011) that quantified author's coding style feature instead of content-based feature by training a set of hidden Markov models. In this way, we can achieve plagiarism detection robust against disguising copied source codes and ghost writing which were both difficult to deal with content-based similarity measurements. Furthermore, there is also a possibility of reducing psychological burdens for both teachers and students through the plagiarism detecting process.

The rest of the paper is organized as follows: we explain about the summarized procedures and characteristics of two different methods in Section 2, explain about structure and main functions provided to teachers and students via web-based GUI in Section 3, and summarized the paper with future works in Section 4.

2. Implemented methods

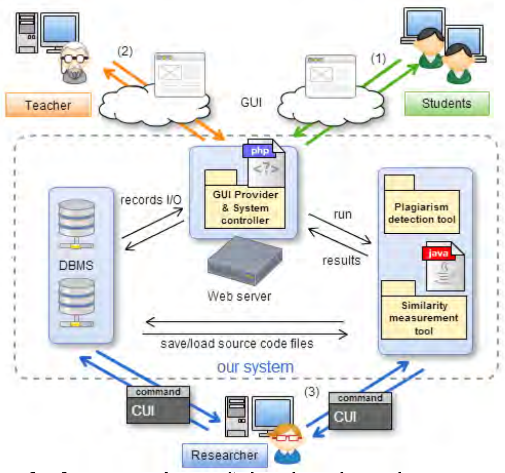
In this study, we developed an online education support system for university programming class. To provide scoring and plagiarism detection function, the system is implementing two kinds of similarity measuring methods: FRef and CM Algorithm. Table 1 summarizes the differences in two methods.

Table 1: Differences in proposed two similarity measuring methods

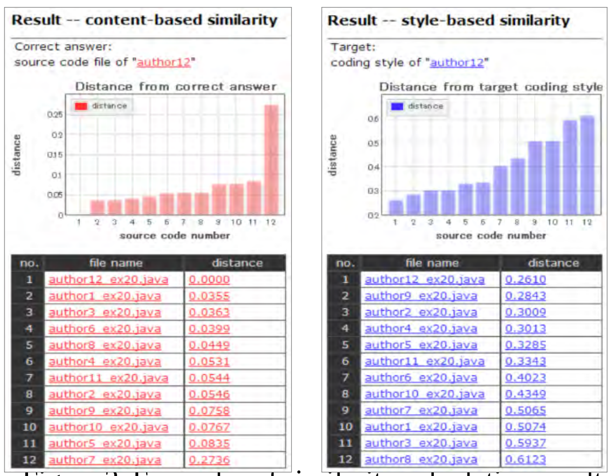
Method	FRef	CM Algorithm
Usage	Scoring, Source code retrieval	Plagiarism detection, Teaching coding standard
How to calculate similarity	Calculate a reference vector with a number of reference source code and calculate Euclidean distance between two source codes' reference vectors	Input a submitted source code and calculate output probability of the coding models (HMM), and compare the result with the model answer's case
Similarity criteria	Content-based similarity	Style-based similarity

FRef is originally proposed to find similar source codes in the repository. As a preprocessing, a source code is firstly tokenized and normalized to be represented as a sequence of tokens. A number of reference source codes had been chosen arbitrary from the repository are also transformed into token sequences. By using two kinds of token sequences, a set of co-occurrence matrices each of which represents distributions of shared tokens between the two sequences are generated. Then the distribution tendency of the shared tokens is quantified by calculating 5 kinds of *Textural Features*. The 5 textural features for each of the co-occurrence matrices are treated as elements of a feature vector called *reference vector*. In this way, content-based feature of a source code is represented relatively by utilizing a number of reference source codes as a number of different bases. The detailed procedure of FRef can be found in the former work (Ohno and Murao, 2007).

As the preprocessing of CM Algorithm, a source code is tokenized and normalized into one of the predefined three kinds of tokens groups: basing point tokens, identification tokens, and others. The tokens used in most source codes regardless the kind of programming languages, such as braces (ex. ``{"), assignment operators (ex. ``="), comments, punctuation marks (ex. ``;") are treated as one of 14 types of *basing-point tokens*. In CM Algorithm, we quantify the superficial feature of source code files occur regardless of the content as the author's *coding style feature*. The feature is defined as the occurrence pattern of *adjacent* tokens of basing point tokens called *identification tokens,* i.e. 1 to 4-letter spaces, tabs, and linefeeds. The remaining tokens are called and normalized as *other tokens*.
After representing a source code by a token sequence consists of one of the three groups of tokens, a token sequence is divided into a number of subsequences by using other tokens as delimiters to fit as input data of coding *models*. The structure and kinds of parameters of the coding models are based on *hidden markov model*. The coding style feature of author is represented as parameters and structures of a set of 28 coding models. Each of which represents different parts of the author's coding style. The detailed procedure of CM Algorithm can be found in the former work (Ohno and Murao, 2011).


Teacher
(2)
GUI
(1)
Students
php
GUI Provider & System controller
records I/O
run
results
Plagiarism detection tool
java
Similarity measurement tool
DBMS
Web server
save/load source code files
our system
command
CUI
(3)
command
CUI
Researcher

igure 1. An overview of the developed system.


Distance from correct answer
distance
source code number
Distance from target coding style

Result -- content-based similarity

Correct answer:
source code file of "author12"

no.	file name	distance
1	author12_ex20.java	0.0000
2	author1_ex20.java	0.0355
3	author3_ex20.java	0.0363
4	author6_ex20.java	0.0399
5	author8_ex20.java	0.0449
6	author4_ex20.java	0.0531
7	author11_ex20.java	0.0544
8	author2_ex20.java	0.0546
9	author9_ex20.java	0.0758
10	author10_ex20.java	0.0767
11	author5_ex20.java	0.0835
12	author7_ex20.java	0.2736

Result -- style-based similarity

Target:
coding style of "author12"

no.	file name	distance
1	author12_ex20.java	0.2610
2	author9_ex20.java	0.2843
3	author2_ex20.java	0.3009
4	author4_ex20.java	0.3013
5	author5_ex20.java	0.3285
6	author11_ex20.java	0.3343
7	author6_ex20.java	0.4023
8	author10_ex20.java	0.4349
9	author7_ex20.java	0.5065
10	author1_ex20.java	0.5074
11	author3_ex20.java	0.5937
12	author8_ex20.java	0.6123

Figure 2. Examples of similarity calculation results.

3. About the system

Figure 1 shows an overview of the system. The system consists of two similarity measurement tools developed in Java, GUI and controller of the system developed in PHP and Database Management System. Figure 2 shows examples of similarity calculation results provided for teachers via web-based GUI. The content-based similarity calculated by FRef is utilized for scoring (left) and the style-based similarity calculated by CM Algorithm is utilized for plagiarism detection (right). As shown in Figure 4, the repository of the system consists two parts: a database utilized in our system and a data set of source code files. A database contains users' information such as ID and password for teachers and students, user profile for students such as his/her coding style feature, progress of assignments, information of assignments such as questions and answers, information of grading such as scores and possibility of plagiarism for each of the assignments. A data set of source code files is only utilized at the preparation phase; that is, calculation of reference vectors for source code similarity measurement and training of coding models to represent students' coding style features utilized for plagiarism detections. Another chance is that when a source code is newly posted, modified, or delated.

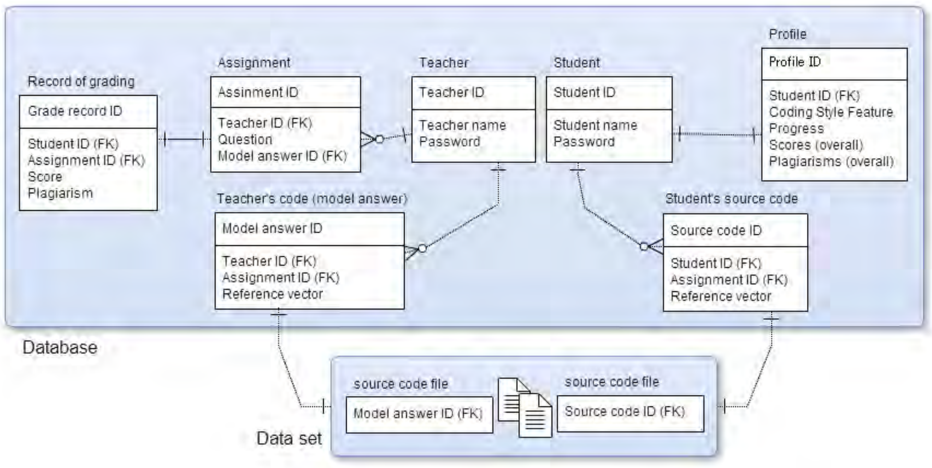


Figure 4. An overview of the repository.

4. Conclusion

In this paper, we developed an online education support system that provides scoring and plagiarism detection functions for the users accessed to the system via web-based GUI. The system is implementing two different kind of similarity measuring method each of which had special features that proved high accuracy in source code similarity measurement according to content- and style-based similarities. As a future work, we apply the system into the real-world programming class for performance evaluation and modification.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B), the grant number is 24700918.

References

Queirós, R. and Leal, J. P. (2012). Programming Exercises Evaluation Systems - An Interoperability Survey. *Proceedings of the 4th International Conference on Computer Supported Education, 1*, 83-90.

Ihantola, P. et. Al (2010) Review of recent systems for automatic assessment of programming assignments, *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pp.86-93.

Ohno, A. and Murao, H. (2007). Measuring source code similarity using reference vectors. *International Journal of Innovative Computing, Information and Control, 3*(3), 525-538.

Ohno, A. and Murao, H. (2011). An Author Identification of In-Class Source Codes by using the Forward-Backward Coding Models. *ICIC Express Letters Part B: Applications*, 2(2), 453-458.