A Source Code Plagiarism Detecting Method Using Sequence Alignment with Abstract Syntax Tree Elements

Hiroshi Kikuchi, Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Japan
Takaaki Goto, Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Japan
Mitsuo Wakatsuki, Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Japan
Tetsuro Nishino, Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Japan

ABSTRACT

Learning to program is an important subject in computer science courses. During programming exercises, plagiarism by copying and pasting can lead to problems for fair evaluation. Some methods of plagiarism detection are currently available, such as sim. However, because sim is easily influenced by changing the identifier or program statement order, it fails to do enough to support plagiarism detection. In this paper, the authors propose a plagiarism detection method which is not influenced by changing the identifier or program statement order. The authors also explain our method’s capabilities by comparing it to the sim plagiarism detector. Furthermore, the authors reveal how our method successfully detects the presence of plagiarism.

Keywords: Alignment, Plagiarism, Programming, Similarity of Source Code

1. INTRODUCTION

Programming education is one of the most important subjects in computer science courses. In programming exercises, plagiarism caused by copying and pasting leads to problems for fair evaluation. In this study, plagiarism is defined as someone handing in a report or documentation as his or her own original work which was, in fact, written or created by someone else.

DOI: 10.4018/IJSI.2015070104
Plagiarism is a serious problem. It is difficult for teachers to detect plagiarism with the huge volume of reports they review and, therefore, they are unable to give fair evaluations. Furthermore, students cannot improve their skills if they plagiarize. In addition, teachers must be able to detect plagiarism in order to prevent dishonesty among their students. However, comparing source code or reports manually is very time-consuming.

Teachers would benefit from automatic plagiarism detection, which would allow them more time to focus on their students and to prepare new materials. It is also important to reveal the capabilities of plagiarism detection systems to discourage cheaters. Therefore, techniques for automatic plagiarism detection play an important role in education.

Some methods of plagiarism detection are currently available, such as sim (Gitchell, 1999). However, because sim is easily influenced by changing the identifier or program statement order, it fails to provide enough support for plagiarism detection.

In this paper, we propose a plagiarism detection method which is not thwarted by changing the identifier or program statement order. We also explain our method’s capabilities by comparing it to sim.

2. BACKGROUND

2.1. Sequence Alignment

Sequence alignment is a method to calculate a correspondence relationship among strings by adding a space or shifting the alphabetic positions. Strings obtained after alignment are also called sequence alignment. The similarity between two strings can be described by a score. In our method, we first obtain tokens by lexical analysis, and then calculate the sequence alignment and compare the obtained similarity scores.

Here is an example. Let s and t be strings. Sometimes the lengths of s and t are different. We insert a gap symbol “-” in order to line up the strings’ lengths. This process is called alignment. Figure 1 illustrates an example of calculating the alignment between “masters” and “stars”.

As described in Figure 1, it is possible to have several patterns of alignment. Usually, sequence alignment includes a minimum number of gap symbols and fewer different characters in the two strings. After obtaining the alignment, we then compare each character in the same position in the two strings. If two characters are the same, we score it as m point. If two characters are different, we score it as d. Moreover, if one or both characters include
Robust Execution of Mobile Activities in Process-Aware Information Systems
www.igi-global.com/article/robust-execution-of-mobile-activities-in-process-aware-information-systems/178564?camid=4v1a

Considerations of Adapting Service-Offering Components to RESTful Architectures
www.igi-global.com/chapter/considerations-adapting-service-offering-components/72222?camid=4v1a