Assignment Copy Detection Using Neuro-genetic Hybrids

Seung-Jin Yang, Yong-Geon Kim, Yung-Keun Kwon, and Byung-Ro Moon

School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{sjyang,dvp,kwon,moon}@soar.snu.ac.kr

Abstract. It is difficult for teachers to find out copies in their program assignments. We propose a method which detects the similarity of assignments. We first transform a program into a sequence of tokens and provide it to an artificial neural network. The neural network is optimized by a hybrid genetic algorithm. Experimental results showed considerable improvement on artificial neural networks.

It is one of the most time-consuming problems for teachers or grading assistants to find out students who copied other program. It is easy to copy other text with various editors. But it is not an easy job to examine the similarity of the programs manually. There are thus many efforts to develop automated copy detection methods.

We want to develop a system that takes a number of programs as input and returns suspicious pairs of programs. In the system the core engine takes two programs as input and returns the degree of suspicion for plagiarism.

We use token counting method on detecting plagiarism. The frequency of each token is determined by the object and programer of the program. We designed the system as follows. We define 40 tokens for the Java language. And we count the frequencies of these tokens and express the frequencies as a vector. We obtain two vectors $(a_1, a_2, \ldots, a_{40})$ and $(b_1, b_2, \ldots, b_{40})$ from the two programs. Here, a_i and b_i match the 40 attributes each corresponding to a token.These two vectors are used for the input of the neural network. There are two versions in this paper. One uses the vector $(a_1, b_1, a_2, b_2, \ldots, a_{40}, b_{40})$ as the input; the other uses $(|a_1 - b_1|, |a_2 - b_2|, \ldots, |a_{40} - b_{40}|)$ as the input. To test the proposed approaches, we prepared two different program assignment sets submitted in past classes. The program sets are used for the training and the test, respectively. Each set was expanded in the following way. Some base programs were selected from each set, intensively checked to see that they are not copies of one another, and distributed to our laboratory fellows. Then, they generated copy programs from the base programs in their own ways. We added the copy programs into the original data set.

Table 1 describes the input data and Table 2 shows the experimental results. In the table, the "candidates" means the number of program pairs that

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2426–2427, 2003.

Table 1. Input Data

Assignment 1 (training data)		Assignment 2 (test data)	
total pairs	copy pairs	total pairs	copy pairs
2000	159	1475	42

Table 2. Results of copy detection (average of 20 trials)

	output			
	candidates	copies	no copies	accuracy
40-input ANN	44.3	19.3	25	43.6
80-input ANN	51.4	16.3	35.1	31.7
40-input hybrid GA	44.7	20.8	23.9	46.5
80-input hybrid GA	48.7	24.2	24.5	49.7

each algorithm outputs as suspicious candidates. And the columns "copies" and "no copies" represent the number of copy pairs and non-copy pairs, respectively, among candidates. The column "accuracy" means the rate of copies over candidates. The numbers 40 and 80 in the row titles correspond to the number of input neurons.

The 80-input hybrid GA (Genetic Algorithm) performed best among them. We can observe an interesting contrast in the results of ANN (Artificial Neural Networks) and hybrid GAs. In case of ANNs, 40-input ANN performed better than the 80-input ANN. In case of hybrid GAs, on the other hand, 80-input hybrid GA performed better than the 40-input hybrid GA. This is strong evidence that shows the gap between the powers of ANNs and hybrid GAs. Although 80 inputs provide more information than 40 inputs, the problem space of ANN optimization might perhaps be too large for the backpropagatoin algorithm. On the contrary, the hybrid GA took advantage of the full information. The 80-input hybrid GA catched on average 24.2 copies among the 42 copies, although the absolute value would vary depending on the degrees of copies.

To the best of our knowledge, this work is the first for the copy detection problem using genetic algorithms combined with neural networks. The result showed considerable performance improvement over ANNs. It also showed that the model of token counting is attractive, and that the GA plays a crucial role in high accuracy detection of copies.

Acknowledgments. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.