International Journal of Engineering Innovation & Research
Volume 4, Issue 1, ISSN: 2277 – 5668

Detecting Backward Students from Code Similarity in Context of Teaching Units

Wataru Nishimoto
email: wnishimoto@de.is.ritsumei.ac.jp

Dinh Thi Dong Phuong
email: phuong@de.is.ritsumei.ac.jp

Hiromitsu Shimakawa
email: simakawa@de.is.ritsumei.ac.jp

Abstract: **In the programming class, teachers need to supervise students as efficiently as they can. Otherwise, they cannot supervise all students. An automatic detection of students who write inappropriate source codes enables teachers to supervise them with higher priority.In this paper, we propose a method to detect backward students. The method calculates the similarity of student codes to model codes in the context of teaching units. The students whose codesare low in the similarity to the model code are regarded as the backward student. The method facilitates teachers to detect students to be supervised more easily.**

Keywords: **Context, E-Learning, Naive Bayse, Programming Education, Teaching Units, Text Book**

1. INTRODUCTION

Program are implemented using knowledge on computers, programming languages, algorithms, and data structures. To implement preferable programs, it is necessary for students unfamiliar with programming to understand not only basic programming knowledge but also programming styles in coding. To make the students understand both of them, many universities provide two kinds of programming classes: lectures and exercises. In lectures, the students study basic programming knowledge, while they try to make program codes for assignments using the basic programming knowledge in exercises. In general, students write individual source codes to submit for the assignments. This paper refers to source codes submitted by students as student codes.

To pick up students to be supervised, teachers checks the student codes, sometimes inquiring questions about them on the students. In exercises classes, only a few teachers supervise a large number of students. The teachers cannot provide supervisions suitable for individual students, unless they efficiently analyze the student codes. In the analysis of individual student codes, they have to grasp which students need supervisions and what kinds of supervisions should be provided. Without efficient analysis, it is hard to supervise all students in fixed school hours. In most of cases, students who understand programming do not need attentive supervision. However, teachers should provide careful supervision for students who do not understand programming, using a lot of time.

The objectives of programming classes are to make students understand basic programming knowledge. Teachers need to find students who do not understand programming, to provide careful supervision so that should make them to understand programming. A efficient tool to find poor students would enable teachers to spend more time on them with higher priority than students superior to them in understanding of programming. It would make whole students in the class understand programming well.

It is easy to find students who cannot write source codes. However, there are many students who write source code satisfying specification of the assignments but write them in inappropriate ways. To find students who write such source codes, teachers should read student codes carefully. Students whose understanding on programming is poor are likely to write source codes only with what they have understood, avoiding trials to achieve new difficult programming concepts. Such negative attitudes for programming training make them write source codes in inappropriate ways. An automatic detection of students who write these kinds of source code would enable, teachers supervise them with higher priority.

This paper proposes a method to detect students who need to be supervised through analysis of the source codes students submit in introductory programming exercise classes. Generally in the programming exercise, teachers provide small assignments containing only a few programming concepts he lectures in a new teaching unit in the textbook, to make students understand them one by one. It means each assignment is strongly related to specific teaching units. Moreover, model codes teachers write for each assignment are compliant to the specific teaching units in the textbook. Suppose related teaching units of a student code are very different from those of a model code, even though they are written for the same assignments. Such student code indicates the student who submits it does not understand what the teacher wants him to understand or use. In other words, the student can be regarded not to understand contents of the teaching units the assignment focuses on.

We use the text classifier to find out student codes related with teaching units identical to those related with model codes. The method proposed in the paper trains a text classifier with the contents of a textbook used in a programming class. The text classifier calculates the similarity of student codes to model codes, in the context of teaching units described in the textbook. The method regards source codes which is low in the similarity as inappropriate ones. It suggests teachers to supervise the students who write the source codes.

The remaining part of the paper is organized as follows. Section 2 describes the current status and problems of programming class in universities. The section also describes students who need to be supervised. Section 3 explains the way to analyze model code and student code, in the context of teaching units of the textbook. Section 4 describes verification of the usefulness of the method, followed by consideration through analysis of assignments

and student codes.

2. Current Situation and Problems in Programming Class

A. Problems in Programming Class

In this paper, we suppose that there is the best programming style for the answer source code of the assignments in programming class. Ideal source codes for assignments in a programming class not only satisfy the specification of the assignments, but also are compliant to the programming style. Programming classes aim to make students understand and acquire basic knowledge of programming, which involves standard input/output, sequence, selection, iteration, function, array, structure, and pointer. The basic knowledge of programming is explained in the sequence of teaching units in a text book, where teachers give the basic knowledge of programming step by step[1]. To improve programming skills, students are required to write source codes compliant to the programming style using the basic knowledge of programming presented in a teaching unit.

Model codes are source codes teachers carefully write to satisfy the specification of assignments with basic knowledge presented in each teaching unit, following the programming styles. Model codes can be regarded as ideal source codes for assignments.

The objective of the programming class is to make students write source codes similar to model codes for themselves. Teachers should analyze student codes, to see whether they meet the two conditions above; they satisfy the specification and use the basic knowledge following the programming styles. Teachers have to carefully readall student codes, to find those which fail to meet the two conditions. If they fail, teachers should decide what kinds of supervision are necessary for the students who write them. Those students need individual supervisions, because causes of the problematic codes vary with the students.

However, a few teachers take care of large number of students in the programming class. To make the matter worse, student codes are hard to read to understand in many cases. Supervision for the students who write problematic codes would take much longer time than that for students who understand programming. Therefore, it is very hard to give adequate supervision for all students. If teachers supervised all students in class, the amount of supervision time for each students would befar from the practical one. It is preferable for teachers to spend enough time to supervise students to be supervised, grasping them beforehand. A method is desired to find students to be supervised before the teachers start individual supervision.

B. Students to be Supervised

Students writing the following source codes need be supervised.

1. The source codes failing to implement functions or methods specified in assignments.

Programs satisfying only input/output specification of assignments do not have to implement all functions and methods indicated by the assignments. But, those kinds of source codes do not fulfill requirements of assignments. But, those kinds of source codes do not fulfill requirements of assignments.

2. Source codes using quite different algorithms from model codes.

There are various ways to write source codes. Programs using statements different from what teachers expect are valid themselves. In the programming class for novice programmers, however, students should learn to make correct use of specified statements rather than make free use of various statements. Programs should be evaluated from not only its right behavior, but also its correct use of specified statements. Unfortunately, students are likely to use statements they have understood, instead of statements expected to be used in the assignments. There are many cases where students write source code without use of expected statements. Students writing those kinds of source codes often fail to understand the expected statements. Teachers should confirm whether those students understand the usage of the statements with specific means such as personal inquiries.

3. Source codes composed of inappropriate codes.

In introductory programming course, new statements and concepts are taught every week. Students who cannot understand them try to solve assignments, using statements and concepts they have understood. The source code written in those statements and concepts would often contain inappropriate codes. Inappropriate codes are grammatically correct, nevertheless they are not recommended to be used from the view point of programming styles. For example, suppose variable ‘date’ is a pointer to a structure representing a specific date in C programming. When students make an access to member ‘month’, in the structure, they should write ‘date->month’. But when they do not understand pointers to structures they often write ‘(*date).month’. This is an inappropriate code. Students are expected to acquire abilities to improve the readability and the usability of the source codes as well as the ability to make programs behave correctly. Although it is overlooked in the first use, it is never recommended to continue to use inappropriate codes. Since inappropriate codes may prevent students from acquiring to write correct codes, teachers should supervise students to stop using inappropriate codes.

The paper refers to students who make source codes above as backward students. Teachers should supervise backward students, because they might be behind other students in understanding of programming.

To find all backward students, teachers should read and understand all of student codes. However, since it is not practical to read all of them, it is very hard to find all backward students. An method is desired to find backward students with automatic evaluation of student codes.

C. Related Works

There are some works to help teachers to supervise students. In web based learning management systems such as [2][3][11], teacher can manage assignments, quizzes, communication with students, and grading students. The work on [11] uses tree view to monitor student progress. The learning management systems liberate teachers from

working on routine jobs. The systems are also beneficial to student to find their understanding in programming. However, they cannot analyze student codes.

Systems called Online Judge automate evaluation of source codes[9]. The online judge systems execute source codes students submit, evaluating the output message of the source codes. However, the online judge systems do not focus on the contents of source codes. It means the systems cannot find inappropriate codes.

There are topics called adaptive e-learning[10]. Adaptive e-learning gives students assignments suitable to their skills or knowledge. It works on an e-learning system. The assignments are decided from the evaluation of assignments students have finished.

For the adaptive e-eaning, the leaning system should be more automated. Otherwise the amount of teacher tasks will be increased so that teachers should supervise students with various type of assignments.

Other works analyze source codes submitted in programming class or their behavior[4][5][13]. The method proposed in [4] analyzes error messages to find understanding levels of students. It provides useful ways to find students who run into deadlocks in coding. The system explained in [5] examines source codes to find plagiarism. The work in [13] analyzes the sequence of function calls, to grasp design abilities of students in programming with a given function library.

However, assignments are easy in introductory programming course. The diversity of source codes arelimited because of the ease of the assignments. It is hard to distinguish plagiarized source codes from others. These methods do not evaluate source code from the both viewpoints of satisfaction of the specification of the assignments and the correct usages of basic programming knowledge presented in each teaching units, which teachers want students to acquire.

3. Detection of Inappropriate Codes Through Text Classification

A. Source Code Related with Teaching Units

Students would study programming, referring a textbook. Contents in a textbook are expected to be well organized. Each of its sections should contain a few pieces of basic knowledge to be taught. This paper considers teaching units correspond to sections in a textbook. We assume each section of the textbook explains one teaching unit students should understand. This paper refers to a set of contents in each section as a teaching unit.

We focus on introductory courses to acquire C programming skills. Any textbook used in introductory programming courses contains illustrations of sentences, exercises along with model codes to use sentences, and descriptions to explain how the model codes satisfy specifications of the exercises with sentences. With these contents, each teaching unit represents a few pieces of basic knowledge of programming students should understand one by one.

Suppose we classify all words in the textbook into teaching units, including natural language descriptions and

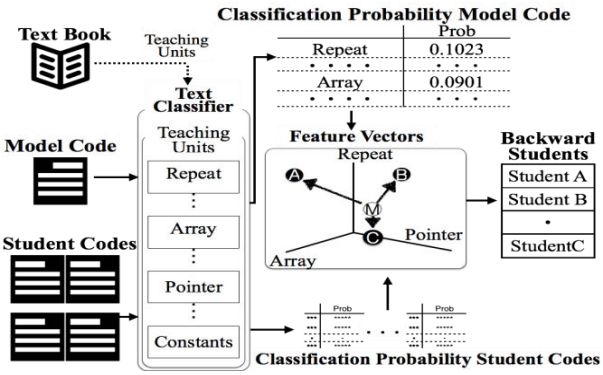

Fig.1. Diagram of Our Method

source codes. Using the naïve Bayes filtering algorithm, a word group corresponding to each teaching unit is provided with a text classifier as training data. After the training, the text classifier is expected to calculate the probability that a sequence of sentences to convey meanings is classified into the each teaching unit, based on the coexistence of words. Let us regard a source code as a sequence of sentences to convey meanings. Given a source code, the trained text classifier calculates the probability which indicates how the source code is related to each teaching unit.

B. Inappropriate Codes in Context of Teaching Units

The paper proposes a method to find backward students using student codes. Fig. 1 shows the diagram of the our method. The method calculates the similarity of student codes to model codes in the context of teaching units in the textbook used in the programming course. Each assignment targets specific teaching units. Teachers write model codes carefully so that they should satisfy specifications of assignments. They are conscious of the compliance of the programming style, when they write the model codes. The low similarity of a student code to the model code for an assignment indicates that it does not follow the teaching unit the assignment targets.

To Illustrates the similarity of model codes and student code, we make a text classifier. To train the text classifier, the method uses all texts including source codes in the textbook. The trained text classifier calculates the probability that a model code for an assignment is related to each teaching unit in the textbook. The result indicates the relevance of the model code to each of teaching units.

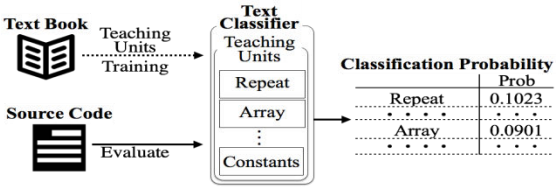

Fig.2. Text Classification

The proposed method takes out teaching units whose probability is more than a certain threshold. The teaching units compose the feature vector of the model code. Here, we refer to the teaching units as the related teaching units of the assignment. Using the text classifier, the method calculates the probability that a specific student code

submitted for the same assignment is related to each of teaching units.

The feature vector of the student code is made of its probability for the related teaching units of the assignments. For each student code, the method calculates the similarity of the feature vector of the student code to that of the model code. The student codes are arranged in the ascending order of the similarity. Students writing the codes low in the similarity have high possibilities to be backward students, who submit insufficient source codes because of the lack of understanding to the related teaching units of the assignment. The method lists up student codes of low similarity. The teachers can find backward students efficiently, inquiring the listed students one by one according to the list order.

C. Text Classifier

We use a text classifier to analyze the model code and the student code. The text classifier implemented using the naive Bayes filtering algorithm[8]. Fig. 2 shows the flow of the text classificaiton and finding probability of the teaching methods. The proposed method trains the text classifier with all words in the textbook written in Japanese. We divide the words in the textbook into groups, each of which corresponds to a section in the textbook. Because Japanese does not have a delimiter like English, we use MeCab[6] to split sentences into words.

Since MeCab is a general morphological analyzer, it is not good at parsing source codes. We make a user dictionary containing words used in C programming language. For example, our dictionary contains “stdio” or “printf”, which MeCab dictionary does not contain. Because of our dictionary, MeCab can split sentences into words more correctly, when it parses contents of the programming textbook.

The proposed method calculates the appearance frequency of each word for every section of the textbook. Note the contents in a section correspond to a teaching unit in the method. It represents characteristics of every teaching unit. We train the text classifier with the appearance frequency of words in each section. Given a

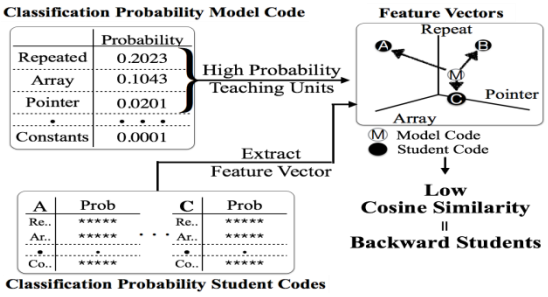

Fig.3. Detect Backward Students

Sequence of sentences, the text classifier calculates the probability that the sequence is classified into eachteaching unit. In this paper, we suppose the probability is the ratio the sequence of sentences is related to each teaching unit. If the sequence of sentences has high relevance to specific teaching units, their characteristics are explained with the contents of the teaching units.

The proposed method treats a source code such as model codes and student codes as a sequence of sentences. It classifies model codes and student codes with the text classifier, to know their relevance to teaching units.

The text classifier can calculates the probability that the model codes are related to the teaching units, because

1) Model codes for assignments written aware of the textbook spontaneously found on the contents of the textbook,

2) Each section of the textbook contains only few teaching units, and

3) A model code for an assignment to acquire programming skills explained in a section is distinctive of related teaching units of the assignment.

If a student code is similar to a model code in the context of teaching units, the student code founds on the same teaching units as the model code.

D. Similarity Between Source Codes

The text classifier calculates the probability that a source code is related to every teaching unit. Suppose a model code written as a solution for a specific assignment. The proposed method focuses on teaching units to which the model code is related with high probability. Fig. 3 shows the flow of detecting backward students. They are the related teaching units of the assignment, which are essential to solve the assignment. The probability for each of the related teaching units composes the feature vector of the model code.

Let us consider a student code for the same assignment. The text classifier calculates the probability that the student code is related to each of teaching units. The feature vector of the student code is also composed of the probability for the related teaching units of the assignment.Suppose the student code founds on teaching units different from the related teaching units of the assignment. It is highly possible that the student code is written in a peculiar way without using the teaching units essential to solve the assignment as expected. It is implied that the student code might fail to implement functions or methods specified in the assignment. Even if it satisfies the specification, it might contain different statements from the model code or inappropriate codes, as it is explained in section 2.2. Because of the discrepancy of the teaching units, the feature vector of such student codes faces to the direction different from that of the model code. The cosine similarity of these feature vectors gets low.

The proposed method derives the feature vector for each student code. It calculates the cosine similarity of the feature vector of each student code to that of the model code. If the similarity is high for a specific student code, the student code has few problems, because it founds on the same teaching units as the model code. On the contrary, the student code which has low similarity is not suitable for the answer of the assignment. Since it may fail to implement functions or methods specified in the assignment, its behavior should be checked for several combinations of input data. Even if the behavior satisfies the specification, it probably contains peculiar codes such as statements different from expected ones or inappropriate codes. The method calculates the median value

of the similarity of all student codes. If the similarity of a specific student code is lower than the median value, the method regards the student submitting it as a backward student.

E. Example

To illustrate the text classifier distinguishes coding ways, let us consider Code A and Code B as a model code and a student code, respectively. Code A and Code B are described in Fig. 4 and Fig. 5. Both of them take the same behavior, printing numbers from 0 to 9 on the standard output. To implement the behavior, Code A uses the" statement to specify the explicit ten times loop, while Code B uses number" statement for infinite loop. Code B implements the ten times loop in an implicit way, terminating the infinite loop with the it way.

It is impossible to find the difference between these two codes without reading the source codes. Table 1 and Table 2 present the classification probability of the source codes to the teaching units of a textbook. The teaching units arranged in the descending order of the probability are "Repeated sentence", "How to use of the array", and "Mechanism of loop operation" in Code A, while "Repeated sentence", "Divide according to the number", and "Mechanism of loop operation" in Code B. Code B getting out of the loop with "if" statement is evaluated to be strongly related to teaching unit "Divide according to the number", which has no effect on Code A. Even though the model code and the student code takes the same behavior, the probability of teaching units makes the difference in their coding ways apparent.

This textbook often uses variable "i" as an index of an array. It also uses loop structures to handle each element in an array. Because of it, the both codes using the loop structure and variable "i" are evaluated to be related to teaching units explaining programming matters using arrays with high probability, though they do not use any array. To determine the relevance of source codes to teaching units, it is considered to play a vital role that students use frequent word sequences in the textbook. As the example shows, the relevance of the two source codes to teaching units turns out from small difference of their coding ways such as usage of statements and word sequences.

Code A

```
#include <stdio.h>

int main(void){
 int i = 0;

 for(i=0;i<10; i++){
    printf(rintf(0i);
 }
 return 0;
}
```

Fig.4. Code A

Code B

```
#include <stdio.h>

int main(void){
 int i = 0;

 while(1){
    printf(rintf(1i++);
    if(i >= 10)
    break;
 }
 return 0;
}
```

Fig.5. Code B

Table 1. Classification Porbability of Code A

Teaching Units	Probability
Repeated sentence	0.0245
How to use of the array	0.0217
Mechanism of loop operation	0.0215
Array of structures	0.0205
Delivery of information by argument	0.0197
Character string processing function	0.0197
Relation between array and pointer	0.0195
Use the pointer variable	0.0189

Table 2. Classification Porbability of Code B

Teaching Units	Probability
Repeated sentence	0.022
Divide according to the number	0.0207
Mechanism of loop operation	0.0205
How to use of the array	0.0204
Input check	0.0197
Array of structures	0.0193
Character string processing function	0.0193
Divide in case of three or more	0.0191

Table3. Chi-squread Test Difference of two groups

Assignments	X^2	p-value
No.1	7.49	0.0062
No.2	16.92	0
No.3	9	0.0027
No.4	5.27	0.0218
No.5	3.11	0.079
No.6	20.7	0

Table.4 Chi-squread Test Ratio of backwad students

Assignments	X^2	p-value
No.1	4.45	0.0349
No.2	9.33	0.0022
No.3	5.18	0.0228
No.4	10.04	0.0015
No.5	1.97	0.16
No.6	11.46	0.0007

4. EVALUATION

We have evaluated whether the proposed method detects actual backward students. We pay attention to the following points.

1) The median value of the similarity is a proper threshold to distinguish backward students from others.

2) When the whole students are divided into 2 groups with the median value of the similarity, the lower half contains more backward students than the upper half.

We have picked up 6 assignments to evaluate the method.

The 6 assignments are issued to teach strings, functions, arrays, structures, and pointers. These assignments are issued in the later lessens in the one semester prog-ramming course, in which more than 500 students are enrolled in Ritsumeikan University. Though they have diversity in the degree, they are difficult assignments for students compared with ones issued in the former lessons of the course. They are not issued according to their

Table 5. Assignment No.1

		Real	
		Proper	Improper
Detected	Proper	33	4
	Improper	21	15

Table6. Assignment No.2

		Real	
		Proper	Improper
Detected	Proper	30	7
	Improper	11	25

Table7. Assignment No.3

		Real	
		Proper	Improper
Detected	Proper	30	7
	Improper	16	20

difficulty. Each student submits one source code for every assignment. We have evaluated 73 student codes for each assignment.

We have manually evaluated them before we apply the proposed method. In the manual evaluation, we do not care the program output of student codes, except apparent lack of required functionality. If the followings are all satisfied, the student code is marked as a proper code, otherwise marked as an improper code. Students submitting improper codes are regarded as backward students in the manual evaluation.

1) The student code corresponds to the assignment, without mistakes in submission of source codes for other assignments.

2) The student code satisfies all of the specifications of the assignment.

3) The student code uses the same statements used in the model code.

4) The student code does not contain any inappropriate code.

The first item is necessary because some students would submit source codes for another assignment. It means the proposed method has to discriminate student codes made for other assignments. It is impossible to know whether students using different statements from the model code have understood the teaching units the model codes founds. Teachers should inquire them to confirm their understanding. Depending on results of the inquiry, they should supervise the students. There are inappropriate codes which are not recommended to use even though they are grammatically correct. Detecting the source codes containing inappropriate codes, teachers should supervise the students who write the source codes.

A textbook called “Worried C”[7] is used as an example of the textbook. The text classifier classifies every source code in the context of 57 sections of this textbook. Our method provides the list of students arranged with the similarity of their source code to the model code for every assignment. We divide the list into 2 parts by the median value of the similarity. We refer to the lower half of the list as a group of backwards.

If the two groups of the list are independent and the group of backwards contains more backward student than the other, we can suppose that the similarity of student codes to the model code works properly as a measure to detect backward students.

Using Chi-squared test, we validate that the rate of backward students is different in the lower half and the upper half. The null hypothesis is that two group of list does not have any difference. As the result shows in Table.3, the p value is lower than 5%, except Assignment No. 5. Therefore, the null hypothesis is rejected. Hence, the two groups of lists divided by the median value of the similarity are independent in most of the assignments.

Table 8. Assignment No.4

		Real	
		Proper	Improper
Detected	Proper	25	12
	Improper	10	26

Table 9. Assignment No.5

		Real	
		Proper	Improper
Detected	Proper	26	11
	Improper	17	19

Table10. Assignment No.6

		Real	
		Proper	Improper
Detected	Proper	18	19
	Improper	0	36

Next, we validate that the group of backwards contains backward students with higher probability than the rate of backward students in the total list. Note that we know the number of proper student codes and improper studentcodes for each assignment, through the manual evaluation. The null hypothesis here is that the ratio of backward students is same in both of the total list of students and the group of backwards. As the result shows in Table.4, the p value is lower than 5%, except Assignment No. 5. Therefore, the null hypothesis is rejected. As a result, the group of backwards contains backward students with higher probability than the total list.

Next, we have figured out the true positive (TP), the false positive (FP), the false negative (FN), and the true negative (TN) of the group of backwards. TP indicates the number of the student codes computed to fall into the group of backwards as well as determined to be improper in the manual evaluation. FP indicates the number of student codes computed as backwards but determined to be proper in the manual evaluation. FN indicates the number of student codes computed as out of the group of backwards, but determined to be improper. TN indicates the number of student codes computed out of the group of backwards as well as determined to be proper. TP,TN,FN,FP are shown in from Table. 5, Table. 6,Table. 7,Table. 8,Table. 9, Table. 10.

Recall rate R is the rate of student codes computed to fall into the group of backwards among improper student codes. It is calculated with TP/(TP+FN). Precision rate P is the rate of improper student codes among student codes

computed to fall into the group of backwards. It is calculated with TP/(TP+FP). The f-measure f is the harmonic mean of P and R, which is derived with f = 2PR / (P + R). The recall rate, the precision rate, and the f-

Table 11. Recall, Precision and F-Measure

Assignments	Recall	Precision	F-Measure
No.1	0.61	0.89	0.73
No.2	0.73	0.81	0.77
No.3	0.65	0.81	0.72
No.4	0.71	0.68	0.69
No.5	0.6	0.7	0.65
No.6	1	0.48	0.65
Mean	0.72	0.73	0.7

Measure for each assignment are shown in Table. 11. The table shows that they are high enough, the mean of the recall rate, the precision rate, and the f-measure are 0.72, 0.73, and 0.70, respectively.

The reason that the results of Chi-squared test is very low for Assignment No. 5 is that the assignment requires many requirements to satisfy it, compared with others. Students failing to read the specification carefully write source codes lacking the full satisfaction of the requirements. The source codes are marked improper, even if they behave correctly and very similar to the model code. On the contrary, some backward students write source codes similar to the model code because of the detailed instructions of requirements. It reduces the difference of students near the median value.

The proposed method adopts the median value of the similarity as the threshold of the group of backwards, which leads to the extraction of a half of the students as backward students. The precision rate gets low if there are many proper source codes for a specific assignment. Therefore, the precision rate is quite low in Assignment No. 6. Recall is not high except Assignment No. 6. The reason is that there are plenty of student codes which are similar to the model codes, but whose behavior do not satisfy the specifications of the assignments.

5. CONCLUSION

In this paper, we have proposed a method to detect backward students. The method calculates the similarity of student codes to model codes in the context of teaching units in the textbook used in the programming course. To calculates the similarity of codes, we made a text classifier trained with the textbook. The low similarity of a student code to the model code for an assignment indicates that it does not follow the teaching unit of the assignment. The method regards the author of the source codes of low similarity as the backward students.

We have evaluated this method, analyzing more than 400 source codes. In most of cases, the method finds backward students properly. Teachers are able to supervise backward studentswith high priority, for the efficient supervision.

REFERENCES

[1] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, “Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science,”ACM, New York, NY, USA, pp. 450.

[2] WebCT WebCT Course Management System, Lynnfield, MA, WebCT, Inc, 2002, Available: http://www.webct.com

[3] Brusilovsky, P. and Miller, P. “Course Delivery Systems for the Virtual University, Elsevier Science,” Amsterdam, 2001, pp. 167–206.

[4] M. Hristova, A. Misra, M. Rutter, and R. Mercuri, “Identifying and correcting java programming errors for introductory computer science students”, SIGCSE ’03: Proceedings of the 34th SIGCSE technical symposium on Computer science education, 2003, pp.153-156.

[5] Francisco Rosales, Antonio García, Santiago Rodríguez, José L. Pedraza, Rafael Méndez, and Manuel M. Nieto,” Detection of Plagiarism in Programming Assignments,”Education, IEEE Trans. Vol. 51, Issue 2.

[6] Taku Kudo, Kaoru Yamamoto, Yuji Matsumoto,”Applying Conditional Random Fields to Japanese Morphological Analysis,” Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp.230-237.

[7] MMGames, Worried C, Available: http://9cguide.appspot.com/en/index.html

[8] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., “Scikit-learn: Machine Learning in Python”,Journal of Machine Learning Research, 2011, Vol. 12, pp.2825—2830.

[9] Brenda Cheang, Andy Kurnia, Andrew Lim, Wee-Chong Oon, On automated grading of programming assignments in an academic institution, Computers & Education, Volume 41, Issue 2, September 2003, Pages 121-131

[10] Paramythis, Alexandros, and Susanne Loidl-Reisinger. Adaptive learning environments and e-learning standards. Second European Conference on e-Learning. 2003. pp.369-379

[11] Yoo, Jungsoon, et al. "Student progress monitoring tool using treeview." ACM SIGCSE Bulletin. Vol. 38. No. 1. ACM, 2006, pp.373-377

[12] Mödritscher, Felix, Victor Manuel Garcia-Barrios, and Christian Gütl. "The Past, the Present and the Future of adaptive E-Learning." Proceedings of ICL 2004 (2004).

[13] Tanigawa, Kohei, Fumiko Harada, and HiromitsuShimakawa. "Detecting learning patterns during exercise from function call logs." International Journal of Advanced Computer Science 1.1 (2011): 30-35.

AUTHOR’S PROFILE

Wataru Nishimoto received B.E from Ritsumeikan University in 2013. He advanced Graduate School of Ritsumeikan University. He engages in the research on data engineering. He is member of IPSJ.

DinhThi Dong Phuong was born in Quang Nam, Vietnam in 1979. She received her bachelor of mathematics and information science in 2001 in Danang Education University and received the master of information engineering in RitsumeikanUniverstiy in 2008 in Japan. From 2009, she has been a PhD candidate at Ritsumekan. Ms. DinhThi Dong Phuong is currently interested in education engineering.

HiromitsuSimakawa received Ph.D degree from Kyoto Univ. in 1999. He joined Ritsumeikan Univ. in 2002. Currently, he is a professor in Ritsumeikan Univ. His research interests include data engineering, usability, and integration of psychology with IT. He is a member of IEEE and ACM.