The Detector of Plagiarism in *Autograder* for PHP Programming Languages

Nelita Apriyani[1], Aldy Rialdy Atmadja[2], Rifqi Syamsul Fuadi[3], Yana Aditia Gerhana[4]
{nelita.ap@student.uinsgd.ac.id[1], aldy@if.uinsgd.ac.id[2], rifqi@if.uinsgd.ac.id[3],
yanagerhana@uinsgd.ac.id[4]}

Informatics Department, UIN Sunan Gunung Djati Bandung

Abstract. The detection of plagiarism in the PHP language program code is one of the things that needs to be done especially in the field of education. At this time checking plagiarism is done by checking someone's program code with another program code. Of course this takes a lot of time. This encourages researchers to provide a system that can make plagiarism checking activities computerized. The plagiarism checking method has 5 stages of checking, namely Pre-Processing, Tokenization, Exclusion, Similarity measurement and Final similarity calculation. In the Final similarity calculation process, similarity calculations were carried out using the Sorensen Dice Coefficient method. The results obtained from this study are that the system can classify 5 types of percentages of plagiarism, namely 0% not taking plagiarism, <15 means having a little in common, 15-50% being moderate plagiarism, 50% means approaching plagiarism, 100% being program code the whole is the same.

Keywords: Program Codes, PHP, Plagiarism Detection, Plagiarism Classification.

1 Introduction

Plagiarism program codes is one of dishonesty in academic which is often done by someone in programming [1]. Plagiarism program code has different characteristics from other plagiarism actions, and also acts of plagiarism in the program code is difficult to detect manually because it needs a lot of time to checking the process. In some previous research mentioned some characteristics of plagiarism that are often practiced :

1. Copying all codes in program includes comments, input data and user interfaces.
2. Change the structure of program to the others language programming.
3. Using software that can generate codes in program automatically.
4. Using the previous codes then combined it with the program codes obtained by the others.

A State of Art on Source Code Plagiarism Detection is entitled a research that discusses the criteria for plagiarism in program codes by comparing the advantages and disadvantages of existing plagiarism detection tools [2]. The next research is Review of Source Code Plagiarism Detection in Academia which is this research explains the 5 processes that are passed in the detection of plagiarism[1].

ICONISTECH-1 2019, July 11-12, Bandung, Indonesia

DOI 10.4108/eai.11-7-2019.2298067

Research that discusses about the technicque of n-gram carried out in the separation of text into characters is discussed in research Comparison of N-gram Technique and Rabin Karp in Application of Plagiarism Detection in Indonesian Text Documents. This research explains that the N-gram method has a more detecting process faster than other methods [3].
The next research Anti Plagiarism Application with Algorithm Karp-Rabin At Thesis In Gunadarma University. In this research obtained 5 criteria for plagiarism based on the percentage of text similarity using the rabin-karp method[4].

The main objective in this research is to create a system that can detect plagiarism in PHP programming codes by classifying the percentage of plagiarism that can be integrated on the autograder system[5].
Based on previous autograder system research, autograder systems do not have features or capabilities in detecting program code plagiarism. Therefore, a research Plagiarism Detection was conducted in Autograder for PHP Programming, where this research can improve the existing autograder system.

II. Plagiarism Checked Code

The main processes carried out detecting plagiarism in the program codes :

1. Pre-Processing
   In this process the program code is transformed by removing comment blocks, saving spaces, and removing program code that is not related to the conversation[5].
2. Tokenization
   Decomposition of a text string to get a set of words or tokens is called Tokenization. This step needs to be done in retrieving information from a string that will be processed later.
3. N-gram
   The N-Gram technique is based on separating text into strings of length n starting from a certain position in a text. The next n-gram position is calculated from the actual position shifted according to the given offset. The N-Gram for each string is calculated and then compared one by one.
4. Similarity
   The percentage of the level of similarity between documents after checking the plagiarism of the program code.

III Proposed Solution

In this study the valuation method used to construct plagiarism detections system in PHP autograder (Moghrap) with using six stages of the process of checking in performing its detection is illustrated in the figure below.

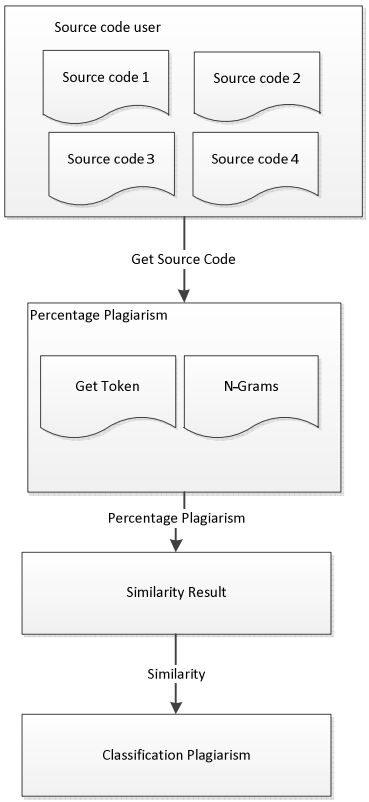

Fig 1. Detection Process of Plagiarism

5. Pre-processing
   In this process the removal process is carried out whitespace, new line (enter), and the symbol of a semicolon.
6. tokenization using PHP Parser At this stage, the process of converting the program code above into token- token contained in php programming language.
7. tokenization with token manual
   From the tokenization process with php parser before they found some token that is not listed in the PHP programming language.
8. Token value conversion
   Below in Table 4.3 is the most tokens each having a value as the initials of the token parser that will constitute the new value of the conversion token.
   Here are some parser php token initialization.

Table 1 Token Initials

TOKEN	INITIALS
T_ARRAY	a
T_BREAK	c

TOKEN	INITIALS
T_CASE	d
T_CLASS	e
T_CLOSE_TAG	f
T_COMMENT	h
T_CURLY_CLOSE	l
T_CURLY_OPEN	m
T_DEFAULT	n
T_DNUMBER	o
T_ECHO	p
T_ELSE	q
T_ELSEIF	r
T_ENDFOR	t
T_ENDFOREACH	u
T_ENDIF	v
T_ENDSWITCH	w
T_ENDWHILE	x
T_EQUAL	y
T_EVAL	z

1. String matching method with N-Grams
   Matching string with N-grams method of function is to check the similarity of the structure of the token code
2. Calculation of Similarity
   At this stage is a stage that serves to calculate the similarity of the program code N-Grams results that have been obtained from the previous process. Here is the formula for calculating the similarity.

$$S = 2\,x\left(\frac{|A \cap B|}{|A| + |B|}\right)$$

   To obtain the results of the formula percentage multiplied by 100%.
   a) Classification Percentage of Plagiarism
      This stage is a stage to determine the categorising of the plagait level. The following is a classification of the level of plagiarism:
      0% : 0% of test results mean both program code is very different both in terms of content and overall.
      < 15% : The 15% test results mean that both program codes have very little in common.
      15 – 50% : The 15-50% test results mean that both program code includes plagiarism level moderate.
      >50% : test results more than 50% mean both program codes include approaching plagiarism.

100%: Test results of 100% mean that both program codes include plagiarism due to the similarity of program code from start to finish.

2 Implementation

The plagiarism detection system on the PHP (mograph) Autograder in the study was implemented into a Web-based system built using PHP native and integrated with the Mograph system. The system has a database that is implemented into the MySQL Mograph. The following on Figure 5 is the architecture of the Mograph system. The following on Figure 6 is the dashboard implementation for the admin page while Figure 7 is an implementation of the problem list page that serves to view the list of questions.

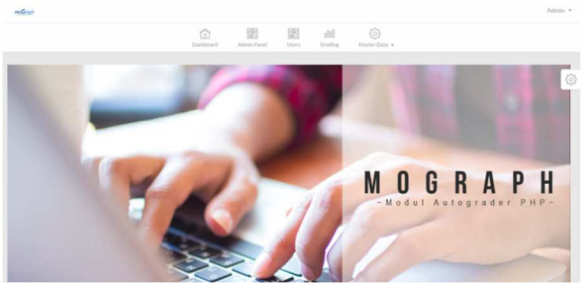

Fig 2. Admin Dashboard Page

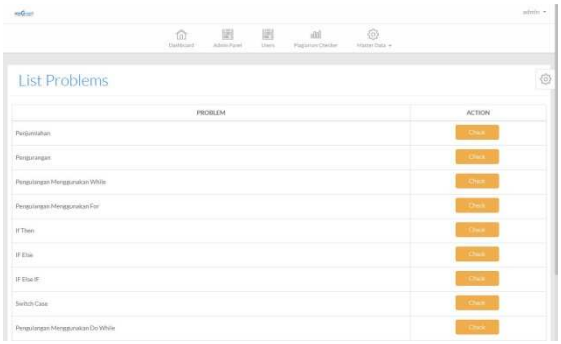

Fig 3. List Problem Page

This problems list page displays the list of problems submitted by the user to check the percentage of plagiarism. Implementation of the page interface of the problems list. On Figure 6 is a checking page of plagiarism.

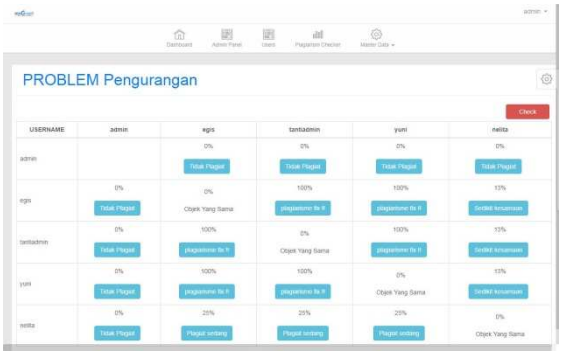

Fig 4. Plagiarism Check page

Implementation of page interface plagiarism checks results is the output of plagiarism checks page.

The next is the implementation of detail page of plagiarism in Figure 7 below.

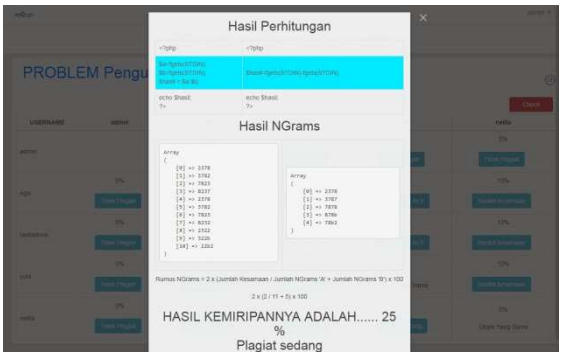

Fig 5 Plagiarism Presentation Detail page

3 Experiment

Testing of the program code's plagiarism checking system was conducted using a black box testing test system, where black box testing was a testing process done by observing the execution results through test data and checking Functional software of the software. The main purpose of this test is to find out whether the system can classify the percentage of plagiarism program code in PHP programming language. Testing was conducted based on 5 categories of percentage classification of plagiarism with test scenarios as follows:

a) experiment comparing 2 pieces of program code with percentage of similarity 100%
b) Experiment showing the percentage of similarity PHP program code
c) The experiments classify the types of plagiarism.

The following are the 2 code programs to be examined percentage of plagiarism.

Table 2 Code Program

CODE PROGRAM 1	CODE PROGRAM 2
<?php $a=fgets(STDIN); $b=fgets(STDIN); $result = $a-$b; echo $result; ?>	<?php $a=fgets(STDIN); $b=fgets(STDIN); $result = $a-$b; echo $result; ?>

The following is an explanation of the classifications of plagiarism experiments in accordance with the methods described in the previous chapter.
From the above code obtained plagiarism check results with the following stages:

1. Pre- processing
   In this process the process of removing whitespace, new line (enter), and the semicolon symbol
2. Tokenization by using PHP Parser
   At this stage the process of converting the program code above into tokens in the PHP programming language. Then the conversion result is in Table 4.1 as follows:

Table 3 Results PHP Parser

CODE PROGRAM 1	CODE PROGRAM 2
["T_OPEN_TAG", "T_VARIABLE", "=", "T_STRING", "(", "T_STRING", ")", ";", "T_VARIABLE", "=", "T_STRING", "(", "T_STRING", ")", ";", "T_VARIABLE","=", "T_VARIABLE","-", "T_VARIABLE", ";", "T_ECHO", "T_VARIABLE", ";", "T_CLOSE_TAG"]	["T_OPEN_TAG", "T_VARIABLE", "=", "T_STRING", "(", "T_STRING", ")", ";", "T_VARIABLE", "=", "T_STRING", "(", "T_STRING", ")", ";", "T_VARIABLE","=", "T_VARIABLE","-", "T_VARIABLE", ";", "T_ECHO", "T_VARIABLE", ";", "T_CLOSE_TAG"]

3. Tokenization with manual tokens
   From the tokenization process with the previous PHP parser still found some tokens that are not listed in the PHP programming language, then re-performed the subsequent tokenization that generates the conversion in the following table:

Table 4 Tokenization with manual tokens

CODE PROGRAM 1	CODE PROGRAM 2
Array (T_VARIABLE T_EQUAL T_STRING T_KURUNG_BUKA T_STRING T_KURUNG_TUTUP T_VARIABLE T_EQUAL T_STRING T_KURUNG_BUKA T_STRING T_KURUNG_TUTUP T_VARIABLE T_EQUAL T_VARIABLE T_MINUS T_VARIABLE T_ECHO T_VARIABLE)	Array (T_VARIABLE T_EQUAL T_STRING T_KURUNG_BUKA T_STRING T_KURUNG_TUTUP T_VARIABLE T_EQUAL T_STRING T_KURUNG_BUKA T_STRING T_KURUNG_TUTUP T_VARIABLE T_EQUAL T_VARIABLE T_MINUS T_VARIABLE T_ECHO T_VARIABLE)

4. Token Value Conversion
   Based on the table above, the tokenization conversion results in initials are as follows:

Table 5 Token Value Conversion Results

CODE PROGRAM 1	CODE PROGRAM 2
SyNANBSyNA NBSySDSpS	SyNANBSyNA NBSySDSpS

5. N-Gram Method
   The result of matching strings with the N-grams method of the function is to check the similarity of the program code 1 token structure with program code 2 based on the initials in the previous table, with the result in the table below:

Table 6 N-Gram Conversion results

CODE PROGRAM 1	CODE PROGRAM 2
Array ([0] => SyNA [1] => yNAN [2] => NANB [3] => ANBS [4] => NBSy [5] => BSyn [6] => SyNA [7] => yNAN [8] => NANB [9] => ANBS [10] => NBSy [11] => BSyS [12] => SySD [13] => ySDS [14] => SDSp [15] => DSpS)	Array ([0] => SyNA [1] => yNAN [2] => NANB [3] => ANBS [4] => NBSy [5] => BSyn [6] => SyNA [7] => yNAN [8] => NANB [9] => ANBS [10] => NBSy [11] => BSyS [12] => SySD [13] => ySDS [14] => SDSp [15] => DSpS)

6. Calculation of Similarity value

   Based on the result of the conversion of N-grams above, there is a similarity of program code 16 tokens (Array 0 – Array 15). Then the calculation is obtained:

$$S = 2 \text{ x} \left(\frac{|A \cap B|}{|A| + |B|} \right)$$

$$S = 2 \text{ x} \left(\frac{16}{16 + 16} \right)$$

$$S = 2 \text{ x } (0{,}5)$$

   To get the percentage then the result is multiplied by 100% then the result is

$$S = 1 \text{ x } 100 \%$$

$$S = 100 \%$$

   From the above calculation, it can be classified that the program code is 100% plagiarism

   The above process is done repeatedly so that it will produce classifications of plagiarism as follows:

Table 7 Result

No	User	Result				
		User1	User2	User3	User4	User5
1	User 1		0%	88%	100%	38%
2	User 2	0%		88%	100%	38%
3	User 3	88%	88%		88%	38%
4	User 4	100%	100%	88%		38%
5	User 5	38%	38%	38%	38%	

4 Conclusion

Once implemented the implementation of 6 plagiarism detection process The program code described in the previous chapter, it can be seen that the system can display the similarity percentage of the program code simultaneously, then can Classify many types of plagiarism in a single time.

References

[1] M. Novak, "Review of source-code plagiarism detection in academia," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings*, 2016.

[2] M. Agrawal and D. K. Sharma, "A state of art on source code plagiarism detection," in *Proceedings on 2016 2nd International Conference on Next Generation Computing Technologies, NGCT 2016*, 2017.

[3] P. N. Technique, D. A. N. Rabin, and K. Pada, "Pendeteksi Plagiarisme Dokumen Teks Bahasa Indonesia," *Semin. Nas. APTIKOM*, vol. 10, no. 28, pp. 753–759, 2016.

[4] A. B. Mutiara and S. Agustina, "Anti Plagiarism Application with Algorithm Karp-Rabin at Thesis in Gunadarma University," Nov. 2008.

[5] W. B. Zulfikar and N. Lukman, "PERBANDINGAN NAIVE BAYES CLASSIFIER DENGAN NEAREST NEIGHBOR UNTUK IDENTIFIKASI PENYAKIT MATA," *J. Online Inform.*, 2016.