Code Comment Assessment Development for Basic Programming Subject using Online Judge

Rosa Ariani Sukamto[1], Rani Megasari[2], Erna Piantari[3], M. Nabillah Fihira Rischa[4]
{rosa.ariani@upi.edu[1], megasari@upi.edu[2], erna.piantari@upi.edu[3]}

Universitas Pendidikan Indonesia, Jl. Setiabudi No. 229 Bandung[1,2,3]

Abstract. The computer program source code must have good documentation on it to be more understandable by others. One of the documentation is code comments. Therefore, university students need to learn and able to write comments. Some university students are familiar with an online judge as a tool for learning programming. An online judge usually does not check how the source code written instead of the output only. This research is aimed at implementing the code comment assessment module for online judge in the order it can check the comments in the source code which is uploaded by students. The comments on this module are categorized into two parts. There are header and process comments. The comments dataset have taken from the online judge which is used before as a learning tool for basic programming subject.

Keywords: Code comment, Comment assessment, Basic programming subject, Online judge, Source code

1 Introduction

Good documentation in source code is an essential thing in the programming field, especially for beginner students in a computer science major. The students need to get used to writing comments in their source code as documentation. The online judge can be used as a tool for programming learning, especially for beginner students. It can collaborate with some learning methods [1]. Code comment assessment for beginner students should be engaged with their learning process and the learning method. There are some research related to code comment assessment such as comment density of open source software which uses comment density as criteria [2], Analyzing the co-evolution of comments and source code which tracks source code entities co-evolution [3], Automatic quality assessment of source code comments: the JavadocMine which focuses on the language of the code comment [4], Quality analysis of source code comments which focuses on code comment coherences [5], A new dataset for source code comment coherence which focuses on the construction of the dataset [6], Deep Code Comment Generation which develops code comment generation tool [7], and Modern code review: a case study at Google which studies about modern code review [8]. Most of those researches are a focus on advanced source code.

This research develops code comment assessment using the dataset from the online judge which is used before. This module should be engaged with beginner student's programming skills and learning processes. The code comment assessment involves comment density, the similarity between comment as header part and header comment from the dataset, and similarity between comments and the given problem. Comment density is chosen as one of the

MSCEIS 2019, October 12, Bandung, Indonesia

DOI 10.4108/eai.12-10-2019.2296547

criteria because it is assumed to be a good predictor of maintainability [2], while similarity with the given problem can be assumed as if the code comment is still related to the given problem.

2 Literature Review and Developing Process

2.1 Online Judge

An online judge is a web-based software that is used for judging the output result from the uploaded source code. One of the online judge software is DOMJudge. DOMJudge is usually used for programming competitions such as ACM ICPC (International Collegiate Programming Contest) [9]. DOMJudge is an open-source software that runs in the LINUX operating system. DOMjudge consists of some standard features as a jury/administrator and the participant. As a jury/administrator, there are features such as participants question clarification, competition management, programming language management, problem management, scoreboard, submission management, participant management. Its standard features as a participant are sent clarification question, submit the submission, and scoreboard (**Figure 1**). An online judge does not check the source code, but only the output result.

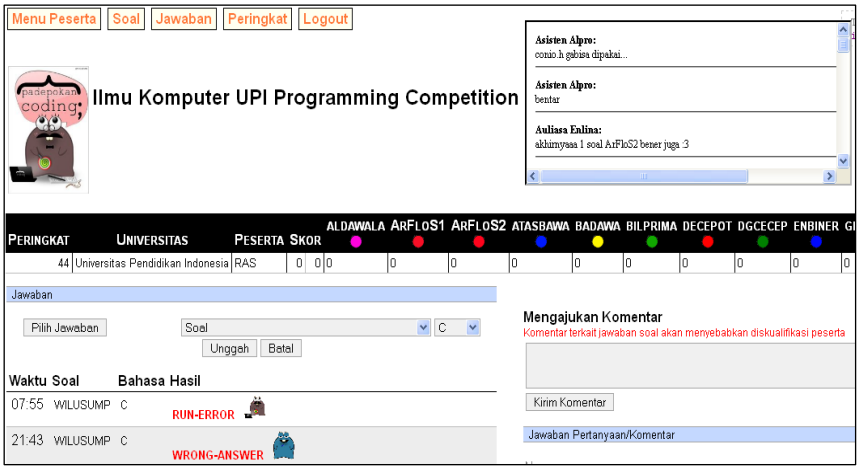

Fig. 1. DOMjudge participant features.

The online judge mentioned in this research is using DOMJudge and has been modified by adding some function and gamification.

2.2 Code Comment Similarity

The string-based similarity is divided into two categories namely character-based and term based. The character-based similarity is such as the LCS method [10]. The implementation of LCS in some words is like:

where the composition of word order may pass through several different characters, but in the order of layout remains the sequence order [11]. In this research, LCS (Longest common subsequence) counts the number of comment words in the source code with the same sequence or sequences and the longest. This research is using statistics and LCS on comment words in the code:

1. Count the number of words from program code comment that is compared to the given problem and it is calculated by the formula (equation 1)[11]:

$$C = \sum_{k \in Dict1 \setminus Dict2} count_1(k) + \sum_{k \in Dict2 \setminus Dict1} count_2(k) + \\ + \sum_{k \in Dict1 \cap Dict2} \max\{count_1(k); count_2(k)\} \quad (1)$$

$$R = \sum_{k \in Dict1 \cap Dict2} \max\{count_1(k); count_2(k)\} \quad (2)$$

$$S = R / C \quad (3)$$

where the count is a function to count the number of existing words in-code comment.

2. Count the number of pairs of adjacent words of two compared and it is calculated using formula (equation 4) [11]:

$$S = \frac{|lp1 \cap lp2|}{\min\{|lp1|, |lp2|\}} \quad (4)$$

where lp1 are a couple of words that exist on the code comment and LP2 are a couple of words that exist on the compared problem.

3. Calculate the number of LCS words on two pieces of code comment and the compared problem or header using formula (equation 5) [11]:

$$S = \frac{|LCS(Word1, Word2)|}{\min\{|Word1|, |Word2|\}} \quad (5)$$

Word1 and word2 come from the two compared where the word is a sentence and the character of the word is a word in a sentence. The similarity from this step is the average of LCS value from every pair compared to sentences that have LCS value more than 0 which is divided by the maximum LCS value.

2.3 Developing Process

The source code dataset is collected from the online judge. The collected data has criteria such as has correct output, and only from basic programming subject with only one file in every given problem. The code comment assessment module in this research is using the similarity values of the statistic method and LCS and the code comment density. Code comment density is the comparison between the number of comment lines and the number of code lines compared with the number of the source code line. Source code will be divided into three parts namely [2]

- A source line of code, or SLoC, is a physical line in a source file that contains source code.
- A comment line, or CL, is a physical line in a source file that represents a comment.
- A line of code, or LoC, is either a source line of code or a comment line.

The developing process has several steps that are represented in **Figure 2**.

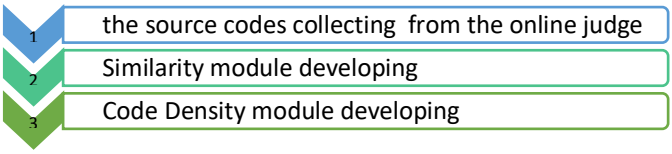

Fig. 2. Developing process.

3 Result and Discussion

3.1 Result

The number of the source code which is used in this research is 100 source code. Some of the results are represented in **Table 1**.

Table 1. The result**.**

code	Comment density	Header	Problem Similarity 1	Problem Similarity 2	Problem Similarity 3
#1	0.02857142857142857	Yes	0.34375	0.0	0.0
#2	0.19148936170212766	No	0.8769230769230769	0.375	0.0
#4	0.5925925925925926	yes	0.5344827586206896	0.16666666666666666	0.0
#5	0.21428571428571427	No	0.2857142857142857	0.1	0.0
#9	0.125	yes	0.1282051282051282	0.0	0.0
#11	0.40425531914893614	yes	0.3644859813084112	0.14705882352941177	0.0
#15	0.14705882352941177	No	0.3076923076923077	0.0	0.0
#18	0.2	No	0.0	0.0	0.0
#23	0.2	yes	0.11764705882352941	0.09090909090909091	0.0
#72	0.06666666666666667	yes	0.0	0.0	0.0

3.2 Discussion

The result shows that beginner students are not used to writing comments on their source code. It can be caused by the limitation of time when they did their evaluation using an online judge. Some source codes are without header comment and very lack of comments, even the only comments are written in the header comment. The higher comment density is only about 0.5. This shows that the beginner students are more concern about the output of their program than how to write the documentation for their source code. According to the results, The LCS method is not suitable for checking the similarity between comment and problem.

4 Conclusion

Code comment assessment in this research is still limited to the density number and similarity. Nevertheless, it can show that beginner student are not used to write comments on their source code. This module will need some adding criteria, especially for the coherence between the source code and the comment.

References

[1] Sukamto, R. A., and Rahman, E. F.: Collaborative online judge and learning methods for learning programming. International Seminar on Mathematics, Science, and Computer Science Education. pp. CS-956 (2015)

[2] Arafat, O. and Riehle, D.: Comment density of Open Source Software. 31st International Conference on Software Engineering - Companion Volume. (2009)

[3] Fluri, B., Würsch, M., Giger, E., and Gall, H. C.: (2009). Analyzing the co-evolution of comments and source code. Software Quality Journal. 17(4). pp. 367-394 (2009)

[4] Khamis, N., Witte, R., and Rilling, J.: Automatic quality assessment of source code comments: the JavadocMiner. In International Conference on Application of Natural Language to Information Systems. pp. 68-79. Springer, Berlin, Heidelberg (2010)

[5] Steidl, D., Hummel, B., and Juergens, E.: Quality analysis of source code comments. In 2013 21st International Conference on Program Comprehension (ICPC). pp. 83-92 (2013)

[6] Corazza, A., Maggio, V., and Scanniello, G.: A new dataset for source code comment coherence. Proceedings CLiC-it 2016 and EVALITA (2016)

[7] Hu, X., Li, G., Xia, X., Lo, D., and Jin, Z.: Deep code comment generation. In Proceedings of the 26th Conference on Program Comprehension. pp. 200-210 (2018)

[8] Sadowski, C., Söderberg, E., Church, L., Sipko, M., and Bacchelli, A.: Modern code review: a case study at google. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice. pp. 181-190 (2018)

[9] DOMJudge. https://www.domjudge.org. (2019)

[10] Gomaa, W. H., & Fahmy, A. A.: A survey of text similarity approaches. International Journal of Computer Applications. 68(13). pp. 13-18 (2013)

[11] Grabowski, S., & Bieniecki, W.: Simple techniques for plagiarism detection in student programming projects. Proc. XIV Konferencja Sieci i Systemy Informatyczne-Teoria, Projekty, Wdrożenia. pp. 225-228 (2006)