Machine Learning and Artificial Intelligence
A.J. Tallón-Ballesteros and C. Chen (Eds.)
© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200776

Detecting Similar Versions of Software by Learning with Logistic Regression on Binary Opcode Information

Hyun-il LIM[1]
Department of Computer Engineering, Kyungnam University, South Korea

Abstract. Logistic regression is widely used in decision problems to classify inputs through training from the previously known training data. In this paper, we propose an approach to detecting similar versions of software by learning with logistic regression on binary opcode information. Because the binary opcode information has detailed information for executing software on an individual machine, the learning from the binary opcode information can provide effective information in detecting similar versions of software. To evaluate the proposed approach, we experiment with two Java applications. The experimental results showed that the proposed logistic regression model can accurately detect similar versions of software after learning from training data. The proposed logistic regression model is expected to be applied in applications for comparing and detecting similar versions of software.

Keywords. Logistic regression, Software analysis, Similar version detection, Binary opcode analysis

1. Introduction

In recent computing environments, software plays an important role in various areas. To support efficient development and management of software, it is required to understand the characteristics of software. Software analysis is an approach to understand the specific characteristics of software. Detecting similar versions of software is one of the basic software analyses to figure out the similarity between versions of software and detect similar ones. The approach has various application areas, such as software similarity analysis [1, 2], code clone detection [3], or malware detection [4].

Machine learning [5] is an approach to generating a model for predicting solutions for given problems by learning previously known training data. Several related works on comparing the similarity of software by using machine learning have been studied, such as linear regression [6] and support vector machine [7]. Besides, deep neural networks have been applied through analyzing the *n*-grams of binary codes [8], common features of binary data [9], or images of binary codes [10].

Logistic regression is one of various machine learning approaches based on a statistical model to describe the relationship between independent and dependent

[1] Corresponding Author, Department of Computer Engineering, Kyungnam University, 7 Kyungnamdaehak-ro, Masanhappo-gu, Changwon, Gyeongsangnam-do 51767, South Korea; E-mail: hilim@kyungnam.ac.kr.

variables. This method is effectively used in binary decision problems for predicting the possibility of events. In this paper, we present an approach to applying binary code information to logistic regression to detect similar versions of software. Because it is not suitable to directly apply software as training data for analyzing and classifying software in logistic regression, it is essential to generate data for representing the features of software as training data. So, we present a method for generating training data of logistic regression for detecting similar versions of software. To evaluate the proposed approach, we experiment and show the experimental results for detecting similar versions in Java applications.

2. Logistic Regression for Software Analysis

Logistic regression [11] is a linear modeling approach to find the relation between the independent variable x and dependent variable y. In binary logistic regression, the dependent variable y has one of two values of 0 and 1 according to whether the predicting event occurred or not. The logistic regression model is generated from the learning of the relationship between the independent variable x and dependent variable y. The dependent variable y stands for the output data for presenting the occurrence of the event. The trained model can predict the possibility of the event for input data. So, in a binary prediction problem, the dependent variable has the value 0 or 1 according to the occurrence of the event. To formulate such a model, the representative logistic model is described as a function $g(x)$, that is continuously increasing between 0 and 1 according to the input variable x as follows:

$$g(x) = \frac{e^x}{1 + e^x}$$

This logistic model function can be used to model the possibility of an event on the input variable x to the result value between 0 and 1.

To design a logistic model for detecting similar versions of software, it is required to represent the features of software as input data of logistic regression. The output data are labeled 0 or 1, depending on whether the input data is for similar versions of software or not. To describe the features of software as input data, we consider the information of binary opcode of software. The binary opcode consists of instructions that are performed to accomplish a task in computing environments. So, the data represent how the task is performed to achieve the goal of the software. As the opcode information describes the characteristics of software at the instruction level, the information is an important criterion for distinguishing different versions of software. In this paper, we design a logistic regression model for detecting similar versions of software through data analyzed from binary opcode information that is derived by comparing the opcode distribution information of software. For example, when there are n types of instructions in software, let the opcode distribution information of two software A and B be $(a_1, a_2, \cdots, a_n)$ and $(b_1, b_2, \cdots, b_n)$, respectively. Then, the distance data comparing the two software A and B is formulated as follows:

$$d(A, B) = (a_1 - b_1, a_2 - b_2, \cdots, a_n - b_n)$$

So, the distance data is used as input data for logistic regression, and the model is trained with 1 or 0 according to the similarity of the two software.

Logistic regression is a method for finding a linear relationship between input and output to model the probability of a certain class. We design a logistic regression model for learning from the distance data of software for detecting similar versions. After the logistic regression model is trained with the distance data and label with 0 or 1 depending on the similarity of data, the model can be used as a classifier for detecting similar versions of software.

3. Designing Logistic Regression for Detecting Similar Versions of Software

In this section, we design the procedure for the logistic regression approach to detecting similar versions of software. Figure 1 shows the procedure for the designed approach. The binary code analyzer is a stage for analyzing the binary opcode from input software. It analyzes the structure of input software and generates the opcode distribution information. The binary opcode information needs to be compared with other data to distinguish similar versions of software. The training data generator compares the opcode information and makes training data set with labels 0 or 1 according to the similarity. After the logistic regression model is trained with the training data set, a logistic regression model is constructed for detecting similar versions of software.

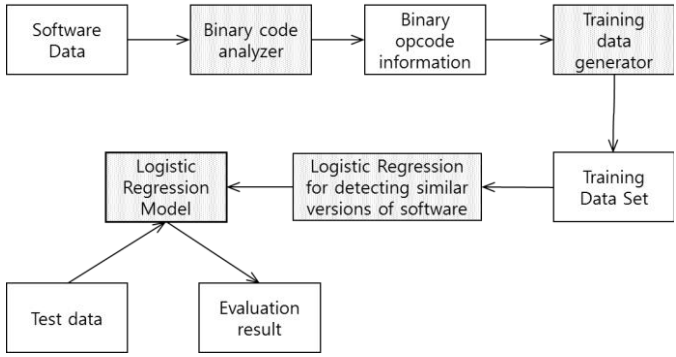

Figure 1. The procedure for applying logistic regression for detecting similar versions of software.

To evaluate the accuracy of the proposed approach, the generated model has experimented with a set of the other test data. The result can show the applicability of the proposed model in detecting similar versions by learning with logistic regression on binary opcode information.

4. Experimental Results

4.1. Experiments

In this section, we experiment on real-world Java applications to evaluate the accuracy of the proposed approach. As described in Figure 1, the binary code analyzer and the training data generator were implemented in Python to analyze binary opcode

information and generate the data set for logistic regression. The logistic regression for the data was implemented in Python and scikit-learn [12] to train the logistic regression model from the generated data and construct a model for detecting similar versions of software.

Table 1. The experimental environment for evaluating the logistic regression model for detecting similar versions of software.

Operating system	CPU	RAM	Binary opcode format
Microsoft Windows 10	Core i7-4790	32GB	Java bytecode

Table 1 shows the experimental environment for evaluating the proposed approach. The experiment was performed in Microsoft Windows 10 operating system with 32 GB of main memory. To evaluate the efficacy of the proposed method, we used Java bytecode as a binary opcode format. Java class files are executable files for Java Virtual Machine and they have Java bytecode as binary opcode to execute programs in the virtual machine. So, Java class files were analyzed to construct data sets of bytecode information. As benchmark software, we used Jakarta ORO and ANTLR for training and evaluation, respectively.

Table 2. The specification of the benchmark software used for training and testing in the experiment.

	Training Data	Test Data
Name of benchmark software	Jakarta ORO	ANTLR
Total # of class files	50	117
Total # of training or test data (similar versions)	1672 (38)	9672 (93)
Max # of bytecodes in a class file	923	1646
Average # of bytecodes in a class file	143.5	172.3
Max # of bytecodes in a similar version	1029	1705
Average # of bytecodes in a similar version	167.7	190.3

Table 2 shows the specification of the benchmark software for this experiment. The information on the numbers of Java bytecodes in benchmarking data is described. To ensure the reliability of the evaluation results, we used the class files that have more than 10 bytecodes as benchmark software. The data sets were generated by analyzing Java bytecode information and comparing the data from two versions of Java class files. The total numbers of data for training and testing were 1672 and 9672, respectively. Among the data, the numbers of data for similar versions were 38 and 93, respectively. To evaluate the detection accuracy for similar versions of software, we used the Smokescreen Java obfuscator to generate similar versions of the benchmark software. The Smokescreen obfuscator generates similar versions by modifying names, Java bytecode instructions, control flows to obfuscate internal structures of the original Java programs. So, the numbers of Java bytecodes in the similar versions were increased as compared to the original versions.

With the training data set from Jakarta ORO, we applied the logistic regression approach to generate a model for detecting similar versions. The generated detection model was evaluated with the test data set generated from ANTLR. From an analysis of

experimental results, we can confirm the applicability of the proposed approach in detecting similar versions of software.

Table 3. The evaluation results of the experiments for detecting similar versions of software with the test data described in Table 2.

	Evaluation Results
Total # of test data (similar versions)	9672 (93)
The # of detections for similar versions	93
The # of false detection for different versions	0
Detection accuracy	100%
Average training time (for 1672 training data)	12.0s

After generating a logistic regression model from the training data, the model was applied to the test data to evaluate how many similar versions of software can correctly be detected. Table 3 shows the evaluation results of the experiment. The number of total test data was 9672, and the number of data for similar versions was 93. The evaluation results showed that all the 93 similar versions in the test data were correctly detected. Besides, there was no false detection for the test data from different versions of software. The average training time was 12.0 seconds, and the detection accuracy was 100% for the test data. These results show that the model trained with the bytecode information of Java class files is highly effective in detecting similar versions of software.

4.2. Discussion and Future Work

From the evaluation results, we confirm that the logistic regression model on Java bytecode information can effectively detect similar versions of software. This is because the learning from the bytecode information can acquire the knowledge needed to distinguish different versions of software by adopting the features in the level of binary instructions. In the proposed approach, we have evaluated the accuracy for detecting similar versions of software with the model trained with the data of the same types of similar versions generated by the Smokescreen obfuscator. Because the training data with the previously known types of similar versions can reflect the exact information needed to detect similar versions of software, the accurate results can be achieved in the evaluation results. So, we confirm that well-prepared training data to reflect the actual environment is important for improving the accuracy of machine learning.

For the practical application of the proposed model for detecting similar versions of software, it needs to generalize the execution environment to adapt data with various types of similar versions. In the evaluation experiments, the model can be trained to fit well to the data of the same types of similarity. In the case where the exact type of similar versions is not clear, it will be difficult to construct accurate training data. In future work, we plan to evaluate the proposed model in environments without knowing the information on similar versions. To improve the ability to detect similar versions of software in real-world environments, we plan to analyze and reflect the additional information from software, such as control flows, opcode sequences, or function calls in the software, as the feature data for identifying the similarity. We also plan to experiment and compare the effectiveness in detecting similar versions of software with other machine learning algorithms.

5. Conclusion

Logistic regression is widely used in binary decision problems. In this paper, we designed an approach to detecting similar versions of software by learning with logistic regression on binary opcode information. Because the binary opcode information has detailed information for executing software, the learning from the information can be applied in modeling logistic regression for detecting similar versions of software. To evaluate the proposed approach, we implemented the bytecode analyzer and logistic regression model and experimented with two Java applications. The experimental results showed that the proposed logistic regression model can accurately detect similar versions of software. We confirm that the binary opcode information is effective data for reflecting the features of software for detecting similar versions. The proposed logistic regression model is expected to be applied as a reliable measure in detecting similar versions of software. Besides, a well-designed model for learning from data is expected to be applied to solve prediction problems in various areas.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Education) (No. NRF-2017R1D1A1B03034769).

References

[1] Niccolo M, Roberto G, Mila DP. A deep learning approach to program similarity. Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis (MASES). September 2018. pp. 26-35.

[2] Noam S, Nimrod P. Binary similarity detection using machine learning. Proceedings of the 13th Workshop on Programming Languages and Analysis for Security (PLAS). January 2018. pp. 42-47.

[3] White M, Tufano M, Vendome C, Poshyvanyk D. Deep learning code fragments for code clone detection. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 2016. pp. 87-98. Singapore.

[4] Danie G, Carles M, Jordi P. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J Network Comp Appl. March 2020. 153(1).

[5] Kevin P M. Machine Learning: A Probabilistic Perspective. The MIT Press. 2012.

[6] Lim HI. A linear regression approach to modeling software characteristics for classifying similar software. Proceedings of COMPSAC, 2019. pp. 942-943.

[7] Lim HI. Applying code vectors for presenting software features in machine learning. Proceedings of COMPSAC. 2018. pp. 803-804.

[8] White M, Tufano M, Vendome C, Poshyvanyk D. Deep learning code fragments for code clone detection. 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 2016. pp. 87-98. Singapore.

[9] Noam S, Nimrod P. Binary similarity detection using machine learning, Proceedings of the 13th Workshop on Programming Languages and Analysis for Security (PLAS). January 2018. pp. 42-47.

[10] Niccolo M, Roberto G, Mila DP. A deep learning approach to program similarity. Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis (MASES). September 2018. pp. 26-35.

[11] Joseph MH. Practical Guide to Logistic Regression. Chapman and Hall. 2015.

[12] Scikit-learn, Machine Learning in Python. https://scikit-learn.org/stable/.