https://doi.org/10.5392/IJoC.2017.13.1.038

Machine Learning Frameworks for Automated Software Testing Tools : A Study

Jungho Kim, Joung Woo Ryu
Analysis & Consulting Team, Big Data Business Division,
ONYCOM Inc., Seoul, 100-101, Republic of Korea

Hyun-Jeong Shin, Jin-Hee Song
School of IT Convergence Engineering/Computer Science & Engineering
Shinhan University, Dongducheon, 483-777, Republic of Korea

ABSTRACT

Increased use of software and complexity of software functions, as well as shortened software quality evaluation periods, have increased the importance and necessity for automation of software testing. Automating software testing by using machine learning not only minimizes errors in manual testing, but also allows a speedier evaluation. Research on machine learning in automated software testing has so far focused on solving special problems with algorithms, leading to difficulties for the software developers and testers, in applying machine learning to software testing automation. This paper, proposes a new machine learning framework for software testing automation through related studies. To maximize the performance of software testing, we analyzed and categorized the machine learning algorithms applicable to each software test phase, including the diverse data that can be used in the algorithms. We believe that our framework allows software developers or testers to choose a machine learning algorithm suitable for their purpose.

***Key words**: Software Testing, Machine Learning, Testing Automation, Software Testing Tool.*

1. INTRODUCTION

According to the survey on global smartphone supply rate in 2015 by the Pew Research Center, Korea indicated the highest smartphone supply rate of 88% among all the subject nations included in the survey [1]. Currently, the mobile apps have been popularizing the mobile devices through providing diverse services that connect on/off-line markets for simple information retrieval, entertainment, shopping, reservation, education and others. The quality and reliability of mobile application software is getting a hot potato as the e-commerce and financial service sensitive to the personal information are provided.

The most important step to developing reliable software is the test. The software testing is labor-intensive, and, therefore, a cost equivalent to half the development resources may be required [1], [2]. The software testing automation is a very effective method that can be used to save costs and decrease manual test errors caused by human through efficiently processing the repetitive and time-consuming test process. In addition, such software testing automation provides the testers with an opportunity to free themselves from the simple and repetitive works to concentrate on more productive and creative works. Since such software testing automation allows the developers to receive feedbacks on their developed results within a short period of time, it is capable of minimizing the development errors.

Due to such reasons, the software testing automation is acknowledged as rather a requirement than an option in the field of software development.

In the software industry, a number of solutions have been released to decrease the repetitive graphic user interface test process [3]-[6]. As the software system became more complicated and the demand for the software testing automation expanded, diverse attempts were made to apply the machine learning-based technologies to the software testing automation. A research on the genetic algorithm-based test case auto production was released [7], [8], and a research predicting the effectiveness of the test case produced based on the artificial neural network was released [9]. A research using the C4.5 decision making tree algorithm to predict potential bugs and locate actual bugs was introduced [10]. In the software test constructed based on various processes, it is impossible to automate all the test processes or to complete all the test processes based on one machine learning algorithm.

* *Corresponding author, Email: jhsong@shinhan.ac.kr*
Manuscript received Jan. 31, 2017; revised Feb. 06, 2017; accepted Feb. 13, 2017

In this paper, we suggest a guideline which can be used by the software testers/beginners who wants to apply machine learning to the software testing automation. The researches relating to the machine learning-based software testing automation are introduced in Chapter 2. We propose the framework, machine learning algorithms and data type categories for enhancing application of machine learning into the software testing automation are proposed in Chapter 3, and the results are covered in Chapter 4.

2. MACHINE LEARNING-BASED SOFTWARE TEST TECHNOLOGY

Researchers [11] introduced the software tests using machine learning techniques, such as C4.5, SVM (Support Vector Machine) and genetic algorithm. Some of the researches on application of C4.5 to test suite evaluation and defective sentence rank assignment are introduced, and some of the SVM-related researches for detecting errors, predicting potentially defective modules, predicting development efforts and optimizing compilers are introduced. Some of the genetic algorithm-related researches for producing test suites with high code coverage, producing test data and removing regression test suites are introduced.

In [12], researchers introduced the software testing for predicting software defects, and they also found that software defects are examined based on the types of machine learning such as classification techniques, grouping techniques, associating rules and hybrid methods.

In [13], the analysis standards are proposed as the framework for analyzing the machine learning-based software testing automation technology. Figure 1 shows the related analysis framework. In [13], six classification methods are proposed as follows.

(1) Classification Based on Test Approach Method

It is a test standard reflecting the level of understanding on the software structure. The test can be classified into black box test, white box test and gray box test [14]. The black box test is used to test externally displayed performances/functions based on the details such as specification. The white box test is conducted based on the source code-level understanding on the internal software structure. The gray test is a complex test conducted through considering both the internal/external system conditions.

(2) Classification Based on Test Activity

The life cycle of the test is classified into test plan, test case management and debugging. For each activity, the sections that can be automated based on machine learning are as follows. For the test plan, the test cost can be predicted through machine learning. For the test case management, the test case priority can be set, the test case can be designed, and the test case can be evaluated through machine leaning. For the debugging, the error location can be detected, the bug priority can be set according to criticality, and the error occurrence can be predicted through machine learning.

(3) Classification Based on Testing Level

The software development process includes diverse development phases ranging from requirements analysis to actualization [15]. The development process must cover software maintenance which is a process after actualization. Accordingly, the testing level can be classified into acceptance test, system test, integration test, module test and regression test.

(4) Classification Based on Learning Technique

In the field of machine learning, diverse learning techniques exist and various characteristics exist as well. According to the machine learning classification by Tom Mitchel [16], machine learning is classified into decision making tree, artificial neural network, genetic algorithm, Bayesian learning, instance-based learning, clustering and hybrid method.

(5) Classification Based on Learning Attribute

Machine learning can be classified according to training data attribute, supervised learning status, time generalization and automation level. The training data can be classified according to data amounts and whether or no noise exists. And the supervised learning status can be classified into supervised and unsupervised. The time generalization consists of online/offline learning statuses which serve as the cumulative learning standards, and can be classified into eager learning mode and lazy learning mode according to the learning time. The level of automation can be classified into partial automation and entire automation.

(6) Classification Based on Learning Factor

Within the software development life cycle or in diverse test phases, the data applicable to the machine learning-based test automation will be generated diversely. The types of data applicable to the software testing automation are software matrix, software specification, control flow graph, call graph, test case, execution data and failure report.

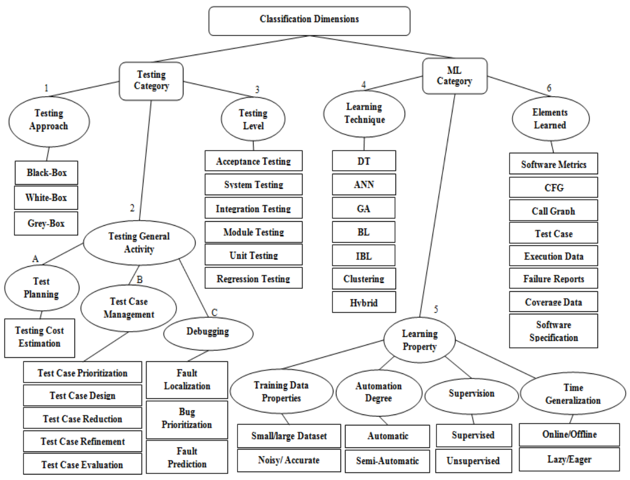

Fig. 1. The Analyzing of Machine Learning-based Software Testing Automation Technology

3. ANALYSIS OF SOFTWARE TESTING TECHNOLOGY FOR APPLICATION OF MACHINE LEARNING

The preexisting analysis report was applicable to several machine-learning algorithms and particular software testing fields. Accordingly, its scope was very limited. It is not easy for the developers and testers unfamiliar with machine learning to apply machine learning algorithms to the software test. Therefore, it is necessary to conduct a research that is capable of providing the developers and testers unfamiliar with machine learning with the baseline data for applying diverse machine learning algorithms to the similar/identical problem solving.

In this thesis, a new machine learning-based framework is proposed as a key to solving this problem. As shown in Table 1, our suggested framework proposes the six standards of test phases, related activities, problem-solving strategies, machine learning-based functions, applicable machine learning algorithms and learning types. The software test phase and related activity define the problems indicated in the software test to which a machine learning algorithm is applied, and the problem-solving strategy signifies the basic approach to the problem solving. Machine learning algorithms and machine learning-based function signify the name/use of the algorithm used for actualizing the basic approach. In addition, the learning type signifies the method used for learning the applied algorithm. Based on such standards, the machine learning-based software testing automation technologies can be systematized as shown in Table 1.

Table 1. A New Machine Learning-based Framework into Software Testing Automation

Test Automation Research	Test Phase	Related Activity	Problem-solving Strategy	Machine Learning-based Function	Machine Learning Algorithm	Learning Type
[17]	Test Planning	Test Cost Estimation	Estimating new test case cost through applying cost for test case group	Test Case Grouping	COBWEB	Unsupervised
[18]		Test Cost Estimation	Estimating new test case cost through applying estimation learning algorithm to test case cost	Value Estimation	Artificial Neural Network	Supervised
[7]	Test Case Management	Test Case Design	Designing highly suitable test case group through test case group evolution process	Test Case Optimization	Genetic Algorithm	Supervised
[19]		Test Case Refinement	Learning/Analyzing expected results per each test case as distribution rule, and, thereby, guiding to detect and refine redundancy of test case	Classification Rule Establishment	C4.5	Supervised
[9]		Test Case Evaluation	Basing test case defect criticality phase on estimation learning algorithm to predict new test case defect criticality phase	Classification Prediction	Artificial Neural Network	Supervised
[20]	Debugging	Fault Localization	Learning code coverage and execution result of test case to predict suspicious code sections	Classification Prediction	Artificial Neural Network	Supervised
[20]		Bug Prioritization	Learning code coverage and execution result of test case to predict and prioritize errors in suspicious code sections	Classification Prediction Result & Related Priority Provision	Artificial Neural Network	Supervised
[21]		Fault Prediction	Learning SW matrix and error number priority per component to predict error number priority based on SW matrix of new component	Classification Prediction	Artificial Neural Network	Supervised
[22]		Fault Prediction	Learning SW update characteristics and bug status to predict update bug status of new software	Classification Prediction	Deep Belief Network	Supervised

Even on solving a similar problem in which the test phase and related activities are relevant, a different algorithm can be used for the software testing automation. The machine learning algorithms and machine learning-based function may vary depending on the problem-solving purpose and strategy. For [17] and [18], the test phase and related activities are identical, but the problem-solving strategy is different. Accordingly, the machine learning algorithms and machine learning-based function required for actualizing the strategy vary. The purpose of [17] is to calculate the cost for testing a new test case based on the cost for testing similar test cases, and the grouping algorithm for detecting similar test cases is used.

[18] is a research on the software matrix of the test case. The machine learning algorithms are used to analyze the numerical relationship between software evaluation indexes and test cost to be used for calculating the cost for testing a new test case. The paper [17] is suitable for estimating the test cost under circumstances where there is not enough machine learning training data, and it proposes similar test cases as the basis for calculating the test cost. The paper [18] provide the complicated correlation between software matrixes and test cost of the test case based on the accuracy of the machine learning algorithm output, so that the stable estimation result can be derived. The paper [7] used the optimization function of a genetic algorithm as a method for decreasing the high test redundancy and increasing the low code coverage which may occur in the test case suite. The paper [19] show a research on the machine learning function used for producing the classification rule according to the simple and implicative attribute value comparison conditions, and it helps the testers prepare a test case which indicates low test redundancy and high code coverage. For [9] and [21], although the test phase and related activities are different, the problem-solving purpose is similar. Accordingly, the identical problem-solving strategy and machine learning-based function are applied. This research compares the similarity between the analyzed characteristics of the test problems and the preexisting research, and, thereby, makes it possible to apply the verified similar problem-solving techniques. In [20], the classification algorithm is used to classify the types of error that may occur in the modules sensitive to risk and cost from the machine learning algorithms. In particular, it contributes to preparation for diverse types of error through rather proposing the likelihood of the all the types of error that may occur than proposing one classified type of error per each module. [22] applies the classification function of the deep belief network included in the deep learning to predict whether or not an error would occur while updating the software version.

The machine learning-based software test research proposed in Table 1 is not the machine learning algorithm that best solves the involved problem. It is possible that other algorithms capable of better solving the involved problem may exist, and the machine learning algorithm is capable of displaying a better performance depending on what input data attribute is selected and how the parameter value is set.

In Table 2, the machine learning algorithms are classified and proposed based on the machine learning-applied standards proposed in this thesis. It is expected that the testers will be able to use Table 2 to apply other machine learning algorithms in order to improve the preexisting research results. In addition, a replaceable candidate for the machine learning algorithm is proposed in preparation for the case where the machine learning algorithm from the preexisting research does not support the type of data relating to the involved problem.

Table 2. Classification of Machine Learning Algorithms Based on Data Types and Functions

Input Data Type / Function	Numeric Type	Nominal Type	Mixed Type (Numeric and Nominal)
Grouping	k-means, ISODATA(Iterative Self-Organizing Data Analysis Technique), DBSCAN(Density-Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points To Identify the Clustering Structure) SOM(Self-Organizing Map)	COBWEB, ROCK(Robust Clustering Algorithm for Categorical Attributes), CLASSIT,	COBWEB/3, EM(Expectation Maximization),
Classification Prediction (Provision of Possibility per Classification Category)	Logistic regression, k-Nearest-neighborhood classifier	ID3(Iterative Dichotomizer 3), Naive Bayesian Classification, Bayesian Belief Network	C4.5, Neural network, Support Vector Machine, Deep Belief Network
Value Prediction	Linear regression, Support vector regression	-	Neural network, CART(Classification and Regression Tree)
Production of Classification Rules	-	ID3(Iterative Dichotomizer 3), Association Rule	C4.5, CART(Classification and Regression Tree)
Optimization	-	-	Genetic Algorithm, Ant colony optimization

The performance of the machine learning algorithm is sensitive to the type/number of learning data, and a better prediction model can be produced through using more types of data. In the software testing, diverse evaluation data are produced per phases, and it is important to select the data and machine learning algorithm suitable for the involved problem. Table 3 is a list of data and types of data which can be produced during the software test process.

The enhanced software testing automation system can be constructed by choosing an appropriate data selection and a machine learning algorithm which can process the data.

Table 3. Classification of Data Applicable to Test Automation Process

Evaluated and Analyzed Data on Software Testing	Description	Type of Data
Error Count per Code Line	• Error count per program source code line	Numeric
Code Coverage	• Source code quantity used for running test	Numeric
Cohesion	• Cohesion level among functional/temporal/procedural similarities	Numeric
Density of Comment	• Ratio between code line count and comment line count	Numeric
Cyclomatic Complexity	• Complexity of codes within one unit(method or function) of software	Numeric
Function Points	• Points quantified into software function units	Numeric
Halstead Complexity	• Software complexity calculated based on operator/operand count within source code (program length/vocabulary count/size/difficulty level/actualized effort count)	Numeric
Instruction Path Length	• Machine instruction count required for execution	Numeric
Class/Interface Count	• Source code class/interface count	Numeric
Code Line Count	• Source code command line count	Numeric
Program Execution Time	• How long program has been executed for	Numeric
Program Loading Time	• Preparation time required for executing program	Numeric
Program Size	• Size of program execution file size	Numeric
Weighted Micro Function Points	• Points for evaluating size of software to be actualized	Numeric
Control Flow Graph	• Graph consisting of nodes displaying basic code blocks and edges displaying flow of code blocks • Used to understand software execution structure such as infinite loop and unattainable code • Example of applied data: node ID/type, edge count per node, cycle count, and path type/count between nodes	Numeric, Nominal
Call Graph	• Graph consisting of nodes displaying unit program(function/procedure/etc.) and edges displaying call relationship between programs • Example of applied data: node ID/type, edge count per node, cycle count, and path route/count between nodes	Numeric, Nominal
GUI Input Event-based Test Case	• Script recording user's mouse/key board event • Example of data: GUI component type, component-related event/input value, and cycle count	Nominal , Numeric
Keyword-based Test Case	• Script saving GUI identifier(keyword)-related event/input value • Example of applied data: GUI component identifier, component-related event/input value, and cycle count	Nominal, Numeric
Image Recognition-based Test Case	• Script saving GUI image and related event/input value • Example of applied data: GUI component identifier, component-related event/input value, and cycle count	Image, Nominal, Numeric

4. CONCLUSION

Testing software is a complex and repetitive task that is time-consuming and costly. In particular, the life cycle of mobile applications is getting shorter and the number of applications using the big data is increasing. All software must be tested with various data to ensure stable use before they are released. Ensuring the reliability of the software is to test all the functionality of the software during the testing process. It is very difficult to test all the functionality of the software in manual. Therefore, the necessity and importance of research on automated software testing are increasing.

The combination of machine Learning and software testing automation enhances the perfection of the software testing and reduces the cost in testing. However, beginners in this field are facing the difficulties on applying machine learning to the software test automation. Most studies at this moment have been confined their work into a certain phase or a certain algorithm of machine learning. Due to these difficulties, it is required to have more various and flexible studies to apply machine learning to the software test automation.

The purpose of our paper is to provide guidelines for novice researchers or testers who want to develop an automated software testing tool using machine learning algorithms. In this paper, the first, we analyzed the case studies of related researches and suggested a new machine learning-capable framework which will help software researchers or testers to more easily apply machine learning to the software test automation. The second, we classified machine learning algorithms according to the data types such as numeric, nominal, and a mixed type. Our contribution will maximize the performance of the machine learning algorithms in software test automation. In specific, our categorization will help software testers to choose the best machine learning algorithm among the usable algorithms. The performance of machine learning algorithms depends on the types and the number of data. In evaluating software quality, it is important to choose the relevant algorithm and the data that is appropriate for the algorithm. Thereby, we analyzed and categorized the types of data that can be used when testing software.

By using our machine learning framework for software testing automation, it is expected that researchers or tester can apply machine learning algorithms in a very flexible manner when they need to solve similar problems given in the software test automation.

ACKNOWLEDGEMENT

This work was supported by the Korea Evaluation Institute of Industrial Technology grant funded by Korea government (Ministry of Trade, Industry and Energy) (NO. ATC 10062957)

REFERENCES

[1] http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/

[2] Boris Beizer, *Software testing techniques* (2nd ed.), Van Nostrand Reinhold Co., 1990.

[3] http://testyd.co

[4] http://www.atam.kr

[5] http://www-03.ibm.com/software/products/en/rtw

[6] http://gareddy.blogspot.kr/2013/01/hp-unified-functional-testing-software.html

[7] Moataz A. Ahmed and Irman Hermadi, "GA-based multiple paths test data generator," Computer & Operations Research, vol. 35, issue 10, 2008, pp. 3107-3124.

[8] Joachim Wegener, Andre Baresel, and Harmen Sthamer, "Evolutionary test environment for automatic structural testing," Information and Software Technology, vol. 43, no. 14, 2001, pp. 841-854.

[9] A. von Mayrhauser, C. Anderson, and R. Mraz, "Using a neural network to predict test case effectiveness," In Aerospace Applications Conference Proceedings, vol. 2, no. 0, 1995, pp. 77-91.

[10] Lionel C. Briand, Yvan Labiche, and Xuetao Liu, "Using machine learning to support debugging with tarantula," In Proceedings of the 18th IEEE International Symposium on Software Reliability, Washington DC, USA, 2007, pp. 137-146.

[11] B. Uma Maheswari and S. Vali, "Survey on Graphical User Interface and Machine Learning Based Testing Techniques," Journal of Artificial Intelligence, vol. 7, no. 3, 2014, pp. 1994-5750.

[12] Pooja Paramshetti and D.A. Phalke, "Survey on Software Defect Prediction Using Machine Learning Techniques," International Journal of Science and Research, 2012, ISSN (Online) pp. 2319-7064.

[13] M. Noorian, E. Bagheri, and W. Du, "Machine learning-based software testing : Towards a classification framework," Proceedings of the International Conference on Software Engineering and Knowledge Engineering, Boston, USA, 2011, pp. 225-229.

[14] Paul Ammann and Jeff Offutt, *Introduction to software testing*, Cambridge University Press, 2008.

[15] Glenford J. Myers and Corey Sandler, *The Art of Software Testing*, John Wiley & Sons, 2004.

[16] Tom M. Mitchell, *Machine learning,* McGraw Hill series in computer science, McGraw-Hill, 1997.

[17] Thomas J. Cheatham, Jungsoon P. Yoo, and Nancy J. Wahl, "Software testing: a machine learning experiment," Proceedings of 23rd annual conference on Computer Science, Tennessee, USA, 1995, pp. 135-141.

[18] Daniel G. e Silva, Mario Jino, and Bruno T. de Abreu, "Machine Learning Methods and Asymmetric Cost Function to Estimate Execution Effort of Software," Proceeding of 2010 Third International Conference on Software Testing, Verification and Validation, Paris, France, 2010, pp. 257-284.

[19] Lionel C. Briand, Yvan Labiche, and Zaheer Bawar, "Using machine learning to refine black-box test specifications and test suites," In Proceedings of the 2008 Eighth International Conference on Quality Software Washington DC, USA, 2008, pp. 135-144.

[20] W. Eric Wong and Yu Qi, "BP neural network-based effective fault localization," International Journal of Software Engineering and Knowledge Engineering, vol. 19, issue 04, 2009, pp. 573-597.

[21] Susan A. Sherer, "Software fault prediction," Journal of Systems and Software, vol. 29, no. 2, 1995, pp. 97-105.

[22] Xinli Yang, David Lo, Xin Xia, Yun Zhang, and Jianling Sun, "Deep Learning for Just-In-Time Defect Prediction," In Proceedings of IEEE International Conference on Software Quality, Reliability and Security, Vancouver, Canada, 2015, pp. 17-26.

Jungho Kim
He received the B.S in computer engineering from Catholic University, South Korea in 2002 and M.S. in computer science from Soongsil University, South Korea in 2005. Currently he is a senior research engineer at the analysis & consulting team, big data business division in ONYCOM Inc. His main research interests include machine learning and distributed computing.

Joung Woo Ryu
He received B.S., M.S., and Ph.D. degrees in computer science from Soongsil University, South Korea. Currently he is a principal research engineer at the analysis & consulting team, big data division in ONYCOM Inc., South Korea. His main research interests include data mining & knowledge discovery, machine learning, soft computing, pattern recognition, and robotics.

Hyun-Jeong Shin
He received the B.S. degree in computer science from Inha University, South Korea, M.S., and Ph.D. degrees in computer science from Soongsil University, South Korea. Currently, He is a professor at School of IT Convergence Engineering, Shinhan University, South Korea. His main research interests include database, system engineering, data mining

Jin-Hee Song
She received B.S. degree in computer science from Seoul National University of Science & Technology, South Korea, M.S. degree in computer science from Hankuk University of Foreign Studies, South Korea, and Ph.D. degree in computer science from Soongsil University, South Korea. Currently, she is a professor at School of IT Convergence Engineering, Shinhan University, South Korea. Her main research interests include parallel algorithms, machine learning, mobile application, and data mining.