International Journal of Science and Engineering Applications
Volume 13-Issue 11, 21 – 25, 2024, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1311.1005

Practice and Application of Intelligent Technology in Software Automation Testing

Zhao Xinyuan
College of Electronic Information and Electrical Engineering
Yangtze University
Jingzhou City, Hubei Province
China

Abstract: As intelligent technology becomes widely adopted and applied, the domain of software testing is experiencing significant transformations, centering around the creation of test data that fulfills particular requirements. Given that traditional testing methods are not only complex and cumbersome, but also have unsatisfactory accuracy and reliability. This article first introduces the basic principles and methods of software testing. Subsequently, the article focuses on introducing and analyzing in depth the latest research progress in software testing based on various intelligent optimization methods. Lastly, this article offers a thorough overview of the present state of automated testing development and anticipates its future directions.

Keywords: Software testing, automated testing, test data generation, intelligence, optimization methods

1. INTRODUCTION

With the rapid development of information technology, software has become an indispensable part of modern society, and its quality and stability are directly related to user experience and business success. In the process of software development, if there is a lack of effective quality assurance, problems and vulnerabilities that arise during use may cause significant losses. Therefore, software testing is crucial in the software development and usage process. However, traditional manual testing is not only time-consuming and inefficient, but also unable to keep up with the pace of software development optimization. Long term tedious and repetitive work can easily demotivate testers, thereby affecting the accuracy of test results. This article aims to summarize the practice and application of intelligent technology in software automation testing, analyze the current situation and challenges in practice, and look forward to future development trends.

2. BASIC CONCEPTS OF SOFTWARE TESTING

2.1 Definition of Software Testing

Software testing is an important component of the software development process, aimed at discovering and evaluating errors, defects, or unexpected functionalities in software [1]. This process requires simulating the entire user operation process and designing and executing specific test cases under each system. Through these interactions, potential errors, defects, or inconsistencies with the design intent in the software can be detected to ensure that it meets the specified requirements and design specifications. By conducting various types of testing, such as unit testing, integration testing, system testing, and acceptance testing, as well as acceptance testing to confirm that the software meets the user's final requirements, the quality of the software is comprehensively evaluated.

Software testing is a continuous iterative process that runs through all stages of software development, from requirement analysis to design, coding, release, and maintenance, playing a crucial role. The procedure for the software is illustrated in Figure 1. Efficient software testing has the potential to substantially decrease the post-release failure rate, enhance user satisfaction, and offer robust assistance for the ongoing enhancement and refinement of the software.

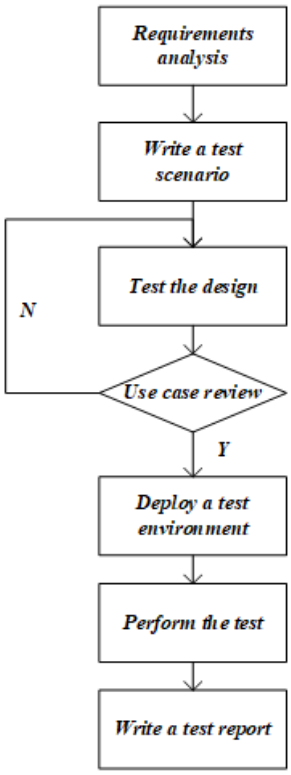

Figure. 1. Software Testing Process Diagram

2.2 Main methods and applications of software testing

The application of artificial intelligence in the field of automated software testing is gradually becoming the key to improving software quality and reliability, and effectively reducing testing cycles and costs [2-3]. The integration of artificial intelligence not only greatly improves the quality and efficiency of detection, but also enables the discovery of problems missed by manual detection. For example, through

International Journal of Science and Engineering Applications
Volume 13-Issue 11, 21 – 25, 2024, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1311.1005

machine learning [4] and pattern recognition techniques, defects discovered during the testing process can be intelligently classified and automatically assigned priority based on their severity and impact range. This helps the development team to respond and fix critical defects faster. Artificial intelligence in software testing mainly includes the following aspects:

1) Test case generation: Automatically generate test cases based on documents or historical data to test various functions of the software. This can gradually reduce time and labor costs [5], and greatly improve the coverage of testing.

2) Performance testing and optimization: Easily simulate large-scale user usage, analyze performance test results, identify performance bottlenecks, which is beneficial for improving system stability and consistency.

3) GUI testing: Simulate user operations, automate complex multi-step tasks, or use image recognition technology to identify and locate various elements in the interface, such as buttons, text, etc.

4) Intelligent testing management: applied to testing management tools, enabling them to have self-learning and optimization capabilities, better adapt to different testing scenarios, unify management, reduce human interference, and improve the work efficiency of testing teams.

3. RESEARCH PROGRESS ON SOFTWARE TESTING BASED ON DIFFERENT INTELLIGENT OPTIMIZATION METHODS

3.1 Automated testing methods based on natural language processing (NLP)

Test cases are a collection of input data used to execute the program under test, and are an important foundation for automated testing. NLP technology plays a core role in machine learning test case generation. Through NLP technology, the system is able to understand, interpret, and generate human language, allowing machines to accurately capture functional requirements and constraints based on the requirements document. For example, when the requirement document mentions that "users should be able to log in by entering their account and password", the NLP system can automatically recognize this requirement and generate corresponding test cases, such as verifying the validity of input boxes, compliance of data, and functionality of buttons. NLP technology can automatically analyze various information sources such as software requirement documents, user feedback, and historical test data, extract key features from them, and automatically generate high-quality test data based on these features [6]. This process not only greatly improves the efficiency of generating test cases, but also reduces human errors, ensuring the accuracy and comprehensiveness of test cases.

Besides creating test cases from requirement documents, NLP can be integrated with various AI technologies to offer more sophisticated approaches to test case generation. For instance, by merging NLP with machine learning models, the test case repository can be significantly enhanced through the incorporation of user feedback and remarks, leading to the production of tailored test cases that assist development teams in promptly identifying and rectifying potential problems. In addition, machine learning also plays a significant role in test result analysis and defect prediction [7-9], such as using hill climbing method, modular annealing method to optimize variable access sequence for fault detection [8], using heuristic strategies to generate fine-grained synchronization sequences, and detecting concurrent faults [9].

3.2 Automated Testing Methods Based on Deep Learning

Deep learning (DL) has been introduced as a subset of machine learning, utilizing more advanced techniques than traditional shallow machine learning techniques, as shown in Figure 2. The concept of neuron and multilayer perceptron (MLP) topology existed before DL was widely adopted. In DL networks, the clever combination of multiple hidden layers and different types of layers such as convolutional layers, pooling layers, and dropout layers together constitute its powerful architecture. The convolutional layer is responsible for automatically extracting key features from input data, such as selecting Convolutional Neural Networks (CNN) [10-12], or combining Recurrent Neural Networks (RNN) with Long Short Term Memory (LSTM) [13]. The pooling layer effectively reduces data dimensionality and preserves important information, while the dropout layer achieves regularization by randomly discarding neurons to prevent overfitting. The introduction of these new architectures collectively promotes DL networks to exhibit better function approximation capabilities in complex tasks.

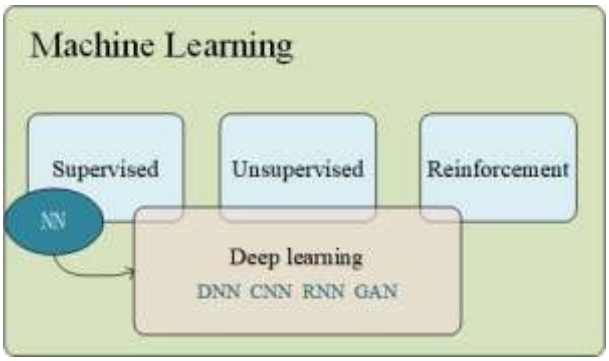

Figure 2. Deep Learning in Machine Learning.

In addition, reinforcement learning has shown great potential in optimizing testing strategies in continuous integration and deployment processes. In this dynamic environment, software undergoes frequent updates, therefore, there is an urgent need for a fast and efficient testing strategy to ensure consistent software quality. Reinforcement learning techniques can assist systems in learning how to accurately select and execute the most critical test cases within limited time resources [14].

3.3 Automated testing methods based on reinforcement learning

Compared to other learning approaches, Reinforcement Learning (RL) excels in discovering optimal decisions within interactive environments, without necessitating a pre-existing dataset to study the interplay between agent and environment. The fundamental workings of the RL process are illustrated in Figure 3. During this process, the agent engages with the environment autonomously as a self-directed learner, progressively refining its strategy. It transitions from a given state in the set of states $s_t \in S$ to select an action from the action space in the set of actions $a_t \in A$. The transition probability between states determines the final state s_{t+1} of the agent. Next, the environment provides a reward r based on the selected action. During this period, the agent continuously

trains until the goal is achieved or the termination condition is met [15].

In reinforcement learning based testing strategy optimization, the testing process is considered as a decision problem. The testing system (agent) interacts with the software (environment), executes test cases, and observes the software's response. If defects are discovered during the testing process, the system will provide a positive reward; If no defects are found, the system will not reward or give a negative reward[16]. In this way, the system can learn how to select and execute test cases to maximize the probability of discovering defects. Alternatively, the coverage of the test can also be used as a reward value to transform the test into a multi-objective optimization problem, in order to find the optimal solution. The data shows that approximately 37% of research reports propose RL based methods that produce better coverage in their respective fields, such as branch coverage and statement coverage[17-18]. Simultaneously, a multitude of research studies suggest that the development of RL models necessitates less time compared to manual creation of test cases, while also attaining superior accuracy and efficiency in their implementation. [19-20].

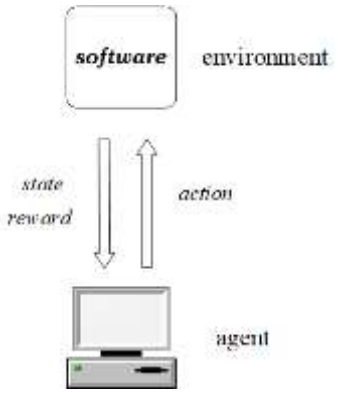

Figure 3. Reinforcement learning cycle.

4. CHALLENGES AND PROSPECTS

Software testing, as a key link in the software development process, has immeasurable value in significantly improving the reliability of software products. The integration of intelligent optimization technology has greatly enhanced the efficiency and quality of software testing, promoting significant progress in this field. However, with the continuous vigorous development of the software industry, software testing technology based on intelligent optimization still faces new challenges and difficulties, and it is urgent for us to continue exploring and innovating to address these emerging issues.

Data quality issues, including accuracy, completeness, and validity of data. Ensure that the test data can truly reflect the real situation and avoid misleading the test results; Ensure comprehensive testing data, covering all possible scenarios and inputs; The test data must comply with business rules and best practices to ensure the effectiveness of the testing. The accuracy and effectiveness of the test results are also a major challenge. Intelligent testing tools require the collection of a large amount of user data for analysis and learning, but data privacy protection has become an important ethical and legal issue. Moreover, for complex business processes and user interactions, automated tools may not be able to completely replace manual judgment. Furthermore, the interpretability of the model is also worth paying attention to. Many machine learning models have strong learning abilities, but they are opaque to workers in the decision-making process, which poses a trust crisis in intelligent testing.

However, intelligent technology will still bring new opportunities for software testing. For example, developing new intelligent tools that can adapt to different testing needs and help development teams analyze and solve problems faster. In addition, cross disciplinary integration is also a major trend, which will bring more possibilities for software testing, such as using big data analysis to optimize strategies.

5. CONCLUSION

Intelligent technology has brought new possibilities for software testing, greatly improving efficiency and reducing the burden and pressure on testers.. This article meticulously examines the practice and application of intelligent technology within the realm of software automation testing, providing an insightful summary of some of the most significant advancements in research methodologies. Furthermore, the article doesn't stop at merely identifying these challenges; it also explores and forecasts the potential opportunities and possibilities that lie ahead.

As the scale and complexity of software continue to grow at an unprecedented rate, software testing methods that rely heavily on intelligent optimization are encountering numerous hurdles that demand immediate attention and resolution. With the relentless advancement of emerging technologies such as machine learning, deep learning, and big data analytics, the field of intelligent technology in software automation testing is poised to exhibit exciting new development trends. These technologies have the potential to revolutionize the way we approach software testing, making it more efficient, accurate, and capable of handling the ever-increasing complexity of modern software systems.

6. REFERENCES

[1] Harman M, Jia Y, Zhang Y Y. 2015 Achievements, open problems and challenges for search based software testing. IEEE 8th International Conference on Software Testing, Verification and Validation.

[2] M. Harman, 2012 The role of artificial intelligence in software engineering, in Proceedings, 1st Int. Workshop on Realizing Artificial Intelligence Synergies Software Engineering, pp. 1–6.

[3] T. Xie, 2013 The synergy of human and artificial intelligence in software engineering, in Proceedings, 2nd International Workshop on Realizing Artificial Intelligence Synergies Software Engineering, pp. 4–6.

[4] M. Noorian, E. Bagheri, and W. Du, 2011 Machine learning-based software testing: Towards a classification framework, in Proceedings, International Conference on Software Engineering and Knowledge Engineering, pp. 225–229.

[5] Michael C C, McGraw G, Schatz M A. 2001 Generating software test data by evolution[J]. IEEE Transactions on Software Engineering,, 27(12): 1085-1110.

[6] Chen T Y, Kuo F C, Merkel R G, et al. 2010 Adaptive random testing: The ART of test case diversity[J]. Journal of Systems and Software, 2010, 83(1): 60-66.

[7] Letko Z. 2010 Sophisticated testing of concurrent programs[C]. The 2nd International Symposium on Search Based Software Engineering. Benevento. 36-39

[8] Bhattacharya N, El-Mahi O, Duclos E, et al. Optimizing threads schedule alignments to expose the interference bug pattern[J]. Search Based Software Engineering.

[9] Qi X F, Zhou H Y. 2019 A splitting strategy for testing concurrent programs[C]. IEEE 26th International Conference on Software Analysis, Evolution and Reengineering. Hangzhou. 388-398

[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, 2012 ImageNet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc.

[11] K. Simonyan, A. Zisserman, 2015 Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.

[12] F. Chollet, Xception: 2017 Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE C onference on Computer Vision and Pattern Recognition, pp. 1251–1258.

[13] S. Hochreiter, J. Schmidhuber, 1997 Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780

[14] Abo-Eleneen, A., Palliyali, A., & Catal, C. 2023. The role of reinforcement learning in software testing. Information and software technology (Dec.), 164.

[15] R.S. Sutton, Dyna, 1991 an integrated architecture for learning, planning, and reacting, ACM Sigart Bull. 2 (4) (1991) 160–163

[16] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, R. Calandra, 2021 On the importance of hyperparameter optimization for model-based reinforcement learning, in: International Conference on Artificial Intelligence and Statistics, PMLR, pp. 4015–4023.

[17] Z. Wu, E. Johnson, W. Yang, O. Bastani, D. Song, J. Peng, T. Xie, 2018 A reinforcement learning based approach to automated testing of android applications, A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-located with FSE (2018) 31–37

[18] D. Adamo, M.K. Khan, S. Koppula, R. Bryce, Reinforcement learning for android GUI testing, in: A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-located with FSE 2018, Association for Computing Machinery, Inc, 2018, pp. 2–8

[19] X. Zhang, M. Lin, D. Zhang, 2012 A learning strategy for software testing optimization based on dynamic programming, in: Proceedings of the Fourth Asia-Pacific Symposium on Internetware, Association for Computing Machinery, New York, NY, USA.

[20] M. Esnaashari, A.H. Damia, 2021 Automation of software test data generation using genetic algorithm and reinforcement learning, Expert Syst. Appl. 183 .

International Journal of Science and Engineering Applications
Volume 13-Issue 11, 21 – 25, 2024, ISSN:- 2319 - 7560
DOI: 10.7753/IJSEA1311.1005