Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Editorial:

Intelligent analysis for software data: research and applications

Tao ZHANG[1], Xiaobing SUN[2], Zibin ZHENG[‡3], Ge LI[4]

[1]*School of Computer Science and Engineering, Macau University of Science and Technology, Macao, China*
[2]*School of Information Engineering, Yangzhou University, Yangzhou 225127, China*
[3]*School of Software Engineering, Sun Yat-sen University, Zhuhai 519082, China*
[4]*School of Computer Science, Peking University, Beijing 100871, China*
E-mail: tazhang@must.edu.mo; xbsun@yzu.edu.cn; zhzibin@mail.sysu.edu.cn; lige@pku.edu.cn

https://doi.org/10.1631/FITEE.2230000

Over the last few decades, software has been one of the primary drivers of economic growth in the world. Human life depends on reliable software; therefore, the software production process (i.e., software design, development, testing, and maintenance) becomes one of the most important factors to ensure the quality of software. During the production process, large amounts of software data (e.g., source code, bug reports, logs, and user reviews) are generated.

With the increase in the complexity of software, how to use software data to improve the performance and efficiency of software production has become a challenge for software developers and researchers. To address this challenge, researchers have used information retrieval, data mining, and machine learning technologies to implement a series of automated tools to improve the efficiency of some important software engineering tasks, such as code search, code summarization, severity/priority prediction, bug localization, and program repair. However, these traditional approaches cannot deeply capture the semantic relations of contextual information and usually ignore the structural information of source code. Therefore, there is still room to improve the performance of these automated software engineering tasks.

The word "intelligent" means that we can use a new generation of artificial intelligence (AI) technologies (e.g., deep learning) to design a series of "smart" automated tools to improve the effectiveness and efficiency of software engineering tasks so that developers' workloads are dramatically reduced.

Currently, advancement has been achieved by a new generation of AI approaches, which are well suited to address software engineering problems. We show two classical and popular automated software engineering tasks using "intelligent" analysis technology for software data as follows:

1. Intelligent software development

Code search and summarization can help developers develop quality software and improve efficiency. Code search is a frequent activity in software development that can help developers find suitable code snippets to complete software projects. Developers usually input the descriptions of these snippets as queries to achieve this purpose. However, it is extremely challenging to design a practically useful code search tool. The previous information retrieval based approaches ignored the semantic relationship between the high-level descriptions expressed by natural language and low-level source code, which affects the performance of code search. Different from information retrieval based methods, deep learning technologies can automatically learn feature representations and build mapping relationships between inputs and outputs. Therefore, the performance of code search is improved. Code summarization is the task of automatically generating natural language descriptions of source code, which can help developers understand and maintain software. In traditional automated code summarization work, researchers tend to use the summary template to extract keywords of source code, which ignores the grammar information of source code. At present, neural network technology has developed vigorously. Convolutional neural networks

[‡] Corresponding author

(CNNs), recurrent neural networks (RNNs), transformers, and other deep learning networks are applied to the task of automated code summarization.

2. Intelligent software maintenance

Severity/Priority prediction can automatically recommend appropriate labels to help developers reduce the workload for labeling severity and priority levels, which are the important features of bug reports. Severity shows the serious levels of the reported bugs, while priority indicates which bugs should be first fixed. The prediction task can help developers quickly assign the important bugs to appropriate developers for fixing them so that the efficiency of software maintenance is improved. Traditional approaches usually adopt machine learning technologies such as support vector machine (SVM) and naive Bayes (NB) to predict the severity/priority level. However, these approaches cannot overcome the problem of data imbalance, so the prediction accuracy is not perfect. Some deep learning technologies, such as CNNs and graph convolutional networks (GCNs), can effectively resolve this problem and capture the contextual semantic information of bug reports so that the prediction performance is improved.

In this context, we organize a special feature in the journal *Frontiers of Information Technology & Electronic Engineering* on intelligent analysis for software data. This special feature covers software architecture recovery, app review analysis, integration testing, software project management, defect prediction, and method rename, as well as related applications. After a rigorous review process, six papers were selected.

Software architecture plays an important role in the software life-cycle, especially in software evolution. It is difficult to maintain up-to-date architecture documentation because it should contain knowledge from all software stakeholders. Therefore, the software architecture recovery task aims to identify and extract architectural information from lower-level representations of a software system such as source code. However, this task is costly in both academia and industry. To resolve this problem, Bixin LI and his collaborators proposed the incremental software architecture recovery technique called ISAR based on existing architectures and related code changes. They built a mapping between code-level changes and architecture-level updates, which can help researchers improve the performance of software recovery technologies. The evaluation results on 10 open-source projects showed that ISAR performs better than selected previous studies.

Identifying emerging topics (e.g., bugs) accurately in user reviews can help developers more effectively update apps. However, the accuracy of emerging topic identification implemented in previous studies is not perfect because user reviews are short and provide limited information. To resolve the problem, Yong WANG and his collaborators proposed an improved emerging topic identification approach. They adopted natural language processing technologies to reduce noisy data and identify emerging topics in app reviews using the adaptive online biterm topic model (AOBTM). Experimental results showed that the proposed approach can effectively identify emerging topics.

Integration testing is an integral part of software testing. Previous studies have focused on reducing test costs in integration testing order generation. However, no studies have explored the testing priorities of critical classes when generating integration testing orders. Hai YU and his collaborators proposed a multilayer dynamic execution network (MDEN) model to quantify the testing priority of each class based on probabilistic risk analysis. In addition, they proposed a strategy to devise an optimal integration testing order, which ensures that high-risk classes can be tested earlier and minimizes the complexity of the test stubs. Experiments on six open-source projects showed that the approach outperforms the baseline approaches for software with different scales.

Cross-project software defect prediction solves the problem of insufficient training data for traditional defect prediction. However, there are still two challenges: (1) Many irrelevant and redundant features in the training process of the model affect the training efficiency and decrease the prediction accuracy; (2) The distribution of metric values varies greatly from project to project due to the development environment, resulting in lower prediction accuracy when the model is applied in cross-project prediction. To address these two challenges, Saihua CAI and his collaborators proposed a new software defect prediction method with metric comprehension based on feature selection and transfer learning. Experimental results showed that the proposed approach shows better performance.

Methods in programs must be accurately named to facilitate source code analysis and comprehension. With the evolution of software, method names may be inconsistent with their implemented method bodies, leading to inaccurate or buggy method names. Many studies have focused on suggesting accurate method names when the method bodies have been modified; however, there are two problems appearing in these studies: (1) a lack of analysis of the structure of method names and (2) a lack of efficient capture of the programming context information. To address these problems, Jingxuan ZHANG and his collaborators proposed a novel method renaming approach to suggest high-quality method names by leveraging structural analysis and lexical analysis. They conducted a series of experiments to validate the effectiveness of the proposed approach, and the results showed that it can significantly improve the state-of-the-art approaches.

Task-oriented virtual assistants are software systems that provide users with a natural language interface to complete domain-specific tasks. However, due to the well-known complexity and difficulty of the natural language understanding problem, it is challenging to manage a task-oriented virtual assistant software project. Shuyue LI and her collaborators shared their practices for addressing the problems and presented the lessons learned at managing a task-oriented virtual assistant software project. They also developed a novel requirement management tool to improve the management efficiency for task-oriented virtual assistant software projects.

Overall, the abovementioned six studies cover many automated tasks to improve the effectiveness and efficiency of software production by analyzing software data. They also provide a series of solutions to overcome the challenges appearing in previous studies. We hope that this collection of interesting and important topics will be beneficial to those with an interest in intelligent analysis for software data or in related areas.

Finally, we would like to express our special gratitude to the authors and reviewers for their support and valuable contributions to this special feature, the editorial staff, and the Editors-in-Chief Profs. Yunhe PAN and Xicheng LU.

Tao ZHANG received his BS degree in automation and ME degree in software engineering from Northeastern University, China, and his PhD degree in computer science from University of Seoul, Korea. After that, he spent one year with the Hong Kong Polytechnic University, China as a postdoctoral research fellow. Currently, he is an associate professor with the School of Computer Science and Engineering, Macau University of Science and Technology (MUST), China. He is a senior member of IEEE and ACM. He has published more than 60 papers in renowned software engineering and security journals such as *IEEE Trans Softw Eng*, *IEEE Trans Inform Forens Sec*, *IEEE Trans Depend Sec Comput*, and *IEEE Softw*, and conferences such as ICSE. His current research interests include mining software repositories and mobile software security.

Xiaobing SUN is a professor in the School of Information Engineering, Yangzhou University, China. He received his BS degree in computer science and technology from Jiangsu University of Science and Technology, China, in 2007. Then, he joined the School of Computer Science & Engineering, Southeast University, China and received his PhD degree in 2012. He has published more than 80 papers in refereed journals (*EMSE*, *STVR*, *IST*, *JSS*, *SCIS*, *FCS*, etc.) and conferences (ICSE, ASE, ICSME, SANER, ICPC, etc.). His research interests include software repository mining and intelligence analysis, software security, etc.

Zibin ZHENG received his PhD degree from the Chinese University of Hong Kong, China, in 2011. He is currently a professor with the School of Software Engineering, Sun Yat-sen University, China. He has published over 150 journal and conference papers, including three ESI highly cited papers. He received the ACM Distinguished Paper Award. His research interests include blockchain, smart contracts, services computing, and software reliability.

Ge LI is an associate professor with the School of Computer Science, Peking University, China. He received his BS degree from Shandong University of Technology, China, in 1999, and his PhD degree from Peking University in 2006. He was a visiting associate professor at the Artificial Intelligence Laboratory of Stanford University, USA, in 2013–2014. He is a deputy general secretary of the CCF Software Engineering Society and the founder of the Software Program Generation Study Group, which includes over 100 advanced researchers in China. His current research concerns mainly applications of probabilistic methods for machine learning, including program language processing, program code generation, and natural language processing.