Research Interests and Representative Contributions

Heng Li, Assistant Professor
Polytechnique Montreal
heng.li@polymtl.ca

Heng Li's research lies within Software Engineering and Computer Systems, with special interests in mining software operational data, software log mining, software performance engineering, monitoring and maintenance of machine learning applications, and mining software repositories. In particular, Heng Li's research primarily focuses on using program analysis, natural language processing, statistical analysis, machine learning, and big data techniques to provide intelligent solutions for software operation challenges (a new emerging area that is called AIOps). Heng Li has been working collaboratively with companies in the software industry (e.g., BlackBerry, Alibaba, and ERA Environmental), researching and developing intelligent approaches to improve the daily maintenance and operations of software and software-intensive systems. The outcomes of his research have brought practical benefits to the industry and inspired follow-up research in related areas. Below, I describe my main research areas and the representative contributions.

1. Understanding and supporting software logging practices

Developers insert logging statements in their source code to monitor the runtime behaviors of software systems. Logging statements print runtime log messages, which play a critical role in various operation and maintenance efforts (e.g., anomaly detection and diagnosing field failures). However, developers typically insert logging statements in an *ad hoc* manner, which usually results in fragile logging code, i.e., insufficient logging in some code snippets and excessive logging in other code snippets. Insufficient logging can significantly increase the difficulty of diagnosing field failures, while excessive logging can cause performance overhead and hide truly important information.

In order to understand and support software logging practices, we surveyed software developers and analyzed software repositories (source code, change history, and issue reports) to understand the benefits and costs of logging, where developers distribute their logging code, how they choose the verbosity level for their logging code, and how they maintain their logging code. Based on the empirical findings, we also proposed automated approaches to support developers' logging decisions, including where to insert logging code, how to choose the verbosity level, and when to update logging code. The findings and approaches make valuable contributions towards improving current logging practices and the quality of logging code.

This line of work has resulted in several papers published in top software engineering journals, including IEEE Transactions on Software Engineering (TSE) [1] and Empirical Software Engineering (EMSE) [2-4]. Although these research papers are relatively new (published between 2017 and 2020), they have received the attention of the software engineering research community (receiving over 100 citations in total) and inspired later research in related areas.

2. Leveraging monitoring data to support the intelligent operations of software systems

Large-scale software systems usually produce a large amount of operational data. The value of such operational data is usually underestimated and rarely leveraged by practitioners. On the other hand, the large size of operational data makes it challenging for practitioners to process. Therefore, we proposed approaches [5-7] to leverage the rich monitoring data of large-scale systems to support intelligent operations.

We proposed an approach that automatically tunes the configuration parameters of a software system to optimize system performance. Our approach leverages the readily available monitoring data (e.g., logs and metrics) to measure the performance of the system in real-time and then optimize the configuration parameters based on the measured performance. Our approach can effectively reduce human intervention on configuration parameter tuning and significantly improve system performance, which has been successfully adopted in a large-scale software system at BlackBerry. This research resulted in a paper [5] that was published in the top conference in the Software Engineering community: The International Conference on Software Engineering (ICSE).

In addition, in order to reduce the losses from node failures in cloud computing platforms, we proposed an AI-based solution that predicts node failures before they actually occur. Our solution leverages readily available alerts and provides actionable suggestions for DevOps engineers to minimize the impact of node failures by performing preventative actions. Our solution is already adopted and used in practice to analyze the monitoring data from tens of thousands of nodes of a cloud computing platform at Alibaba. This work also resulted in a paper published in the ACM Transactions on Software Engineering and Methodology (TOSEM) [6], one of the top journals in Software Engineering.

3. Leveraging monitoring data to support performance assurance of software systems

Performance regressions of large-scale software systems often lead to both financial and reputational losses. In order to detect performance regressions, performance tests are typically conducted in an in-house (non-production) environment using test suites with predefined workloads. However, performance testing is usually very expensive as it requires extensive resources and lasts for an extended period.

We leverage black-box machine learning models to automatically detect performance regressions using the data from field operations of large-scale software systems. Our approach uses black-box models to capture the relationship between the performance of a software system (e.g., CPU usage) under varying workloads and the runtime activities that are recorded in the readily available logs. Then, our approach compares the black-box models derived from the current software version with an earlier version to detect performance regressions between these two versions. Our results show that such black-box models can effectively and timely detect performance regressions in the field. Our approach can complement or even replace typical in-house performance testing when testing resources are limited (e.g., in an agile environment). In addition, we further proposed an approach to automatically locate the root causes of the detected performance regressions. Our approaches have been adopted in practice to detect and address performance regressions of a large-scale software system at ERA Environmental Management Solutions. This line of work led to a paper published in Empirical Software Engineering (EMSE)

[8] and another paper submitted to IEEE Transactions on Software Engineering (TSE), two top journals in Software Engineering.

4. Supporting log analysis through the automated parsing of log data

As logs are usually very large in size, automated log analysis (e.g., automated anomaly detection) is needed to assist practitioners in their software operation and maintenance efforts. Typically, the first step of automated log analysis is log parsing, i.e., converting unstructured raw logs into structured data. However, log parsing is challenging, because logs are produced by static templates in the source code (i.e., logging statements) yet the templates are usually inaccessible when parsing logs. As the volume of logs grows rapidly in the era of cloud computing, efficiency becomes a major concern in log parsing.

Therefore, we proposed Logram, an automated log parsing approach that leverages n-gram dictionaries to parse log data in an efficient manner. Logram outperforms the state-of-the-art log parsing approaches in both accuracy and efficiency. Besides, Logram can effectively support online parsing (i.e., when logs are continuously generated as a stream) with similar parsing results and efficiency as the offline mode. Logram can benefit a wide spectrum of future research and practices that rely on automated log parsing to achieve their log analysis goals. This research was published in the IEEE Transactions on Software Engineering (TSE) [9], one of the top journals in Software Engineering. We also filed a patent for Logram that is in process.

5. Monitoring and maintenance of machine learning applications

Like other software applications, monitoring and maintenance of machine learning (ML) applications are key to ensure the continuous success of ML applications in delivering the accurate results to users or other applications that rely on such ML services. On the other hand, monitoring and maintenance of ML applications have its unique challenges, including that one need to monitor and maintain the data and models in addition to the code. Therefore, this line of research aims to understand the practices of monitoring and maintaining ML applications and address the challenges in such practices.

In an initial study [7], we focus on ML applications in the AIOps (Artificial Intelligence for IT Operations) domain. We observe that AIOps models suffer from concept drift, and periodically updating AIOps models can help mitigate the impact of such concept drift, while the performance benefit and the modeling cost of increasing the update frequency depend largely on the application data and the used models. In a follow-up study [10], we investigated different approaches for maintaining AIOps models, including leveraging statistical analysis to detect concept drift and determine when to update the models, time-based ensemble approaches that combine models trained on smaller time windows, and online learning approaches. We observe that considering sophisticated model maintenance approaches (e.g., applying concept drift detection and using time-based ensembles) could provide better performance and efficiency than simply retraining AIOps models periodically.

References:

[1] H. Li, W. Shang, B. Adams, M. Sayagh, and A. Hassan, “A Qualitative Study of the Benefits and Costs of Logging from Developers’ Perspectives,” IEEE Transactions on Software Engineering, pp. to appear, 2020.

[2] H. Li, T.-H. P. Chen, W. Shang, and A. E. Hassan, “Studying software logging using topic models,” Empirical Software Engineering, vol. 23, no. 5, pp. 2655–2694, 2018.

[3] H. Li, W. Shang, Y. Zou, and A. E. Hassan, “Towards just-in-time suggestions for log changes,” Empirical Software Engineering, vol. 22, no. 4, pp. 1831–1865, 2017.

[4] H. Li, W. Shang, and A. E. Hassan, “Which log level should developers choose for a new logging statement?” Empirical Software Engineering, vol. 22, no. 4, pp. 1684–1716, 2017.

[5] H. Li, T.-H. P. Chen, A. E. Hassan, M. Nasser, and P. Flora, “Adopting Autonomic Computing Capabilities in Existing Large-Scale Systems: An Industrial Experience Report,” in Proceedings of the 40th International Conference on Software Engineering, 2018.

[6] Y. Li, Z. M. Jiang, H. Li, A. E. Hassan, C. He, R. Huang, Z. Zeng, M. Wang, and P. Chen, “Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform: An AIOps Solution,” ACM Transactions on Software Engineering and Methodology, vol. 29, no. 2, pp. 13:1–13:24, 2020.

[7] Y. Lyu, H. Li, M. Sayagh, Z. M. Jiang, and A. E. Hassan, “An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions,” ACM Transactions on Software Engineering and Methodology, pp. to appear, 2021.

[8] L. Liao, J. Chen, H. Li, Y. Zeng, W. Shang, J. Guo, C. Sporea, A. Toma, and S. Sajedi, “Using Black-Box Performance Models to Detect Performance Regressions under Varying Workloads: An Empirical Study,” Empirical Software Engineering, pp. to appear, 2020.

[9] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, “Logram: Efficient Log Parsing Using n-Gram Dictionaries,” IEEE Transactions on Software Engineering, pp. to appear, 2020.

[10] Y. Lyu, H. Li, Z. M. Jiang, and A. E. Hassan, “Assessing the maturity of model maintenance techniques for AIOps solutions,” Under submission.