

C02029: Doctor of Philosophy

33874: PhD Thesis: Software Engineering

Sep 2022

Multi-triage: A Multi-Task Learning Approach to Bug Triageing

Thazin Win Win Aung

School of Computer Science

Faculty of Eng. & IT

University of Technology Sydney

NSW - 2022, Australia

Multi-triage : A Multi-Task Learning approach to Bug Triaging

*A thesis submitted in fulfillment of the requirements
for the degree of*

Doctor of Philosophy
in
Software Engineering

by

Thazin Win Win Aung

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW, Australia

Sep 2022

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Thazin Win Win Aung*, declare that this thesis is submitted in fulfilment of the requirements for the award of the award of Doctor of Philosophy, in the *Faculty of Engineering and IT* at the University of Technology Sydney, Australia.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: _____

[Thazin Win Win Aung]

DATE: 30th Sep, 2022

PLACE: Sydney, Australia

DEDICATION

To my family who brings out the best in me.

To my supervisors for their inspiration and endless support.

To my late partner, my best friend, and my little buddy Albert, for their unconditional love and encouragement.

ABSTRACT

Bug triage plays a significant role in software maintenance activities, including optimization, error correction, and feature enhancement. Triage is the procedure of assigning the severity, issue type, and developer to resolve the issue in the most effective order. Performing triage is time-consuming and challenging, depending on the system's complexity. Thus, it is time-consuming and hinders the effectiveness of linkages between two triage tasks. An automated approach to assisting the issue allocation process to relevant category and developer benefits bug triages. A large body of previous work aims to address the allocation problem by conjecturing the extensive list of approaches ranging from the heuristics-based approach, text retrieval approach, and machine learning approach. However, these studies treated the issue of categorization and assignment tasks as a single task learning model and developed a multiple recommendation system.

This dissertation aims at leveraging the bug triage process by adopting the multi-task learning approach. We developed a multi-triage model, a system for predicting developers and issue kinds for a brand-new issue report. In our approach, we split issue reports, the text description, and code snippets into two separate tokens to conjecture the contributions of each context in the learning model. Furthermore, we addressed the class-unbalanced challenge of data sets by generating synthetic issue reports using the contextual data-augmentation approach.

We conducted four studies in this thesis. The first was an empirical study of the automatic traceability link recovery approach to analyze how previous studies addressed the linkages between software artifacts (e.g., requirements, issue reports, test cases, and source code). The second was the experimental studies about visualizing the linkages of software artifacts using the hierarchical trace map. The first two studies were mainly focused on understanding the broad concepts of how software artifacts can be linked together and presented to stakeholders effectively. Based on this accumulated knowledge, we designed the multi-triage model to identify the linkages between developers, issue types, and issue reports to leverage the bug triage process. Lastly, we conducted a case study of the issue tracking system used by the software consulting company to conjure the process of introducing the automatic developer assignment and labeling recommendation model in the bug triage process.

Our study led to several key findings. We found that the multi-triage model training time and performance are better than single-task learning models. We also uncovered that including the contextual data augmentation-based synthetic bug reports in training data sets can improve the learning model's performance noticeably. Lastly, we presented

the ability to learn issue descriptions and code snippets in two separate tokens and their effect on the learning model.

ACKNOWLEDGMENTS

First and foremost, thank you to my principal supervisor, Professor Yulei Sui, for his mentorship, guidance, and support throughout this long Ph.D. journey and his positive attitude toward life and study encouragement when it was most needed.

Thank you to my kind advisor, Dr. Huan (Angela) Huo and Dr. Yao Wan, for their invaluable lessons, guidance, and encouragement during my studies. Also, I am sincerely grateful to Dr. Christy Jie Liang, who directed me to the brighter side of the research world while I lost my way.

I am extremely grateful to them for their unconditional support, advice, and patience over the years. They opened my eyes to what research in software engineering is all about and taught me how exciting and challenging academia truly is. This achievement would not have been possible without their incredible feedback and mentorship.

I must thank the Australian Postgraduate Research Intern (APR. Intern) industrial internship grant for providing me with funding for the research work. I am also thankful to Dialog IT company for allowing me to conduct a field study in real-world software projects for my thesis writing. I would also like to thank all the software consultants involved in the field study for this research project. The field study could not have been successfully conducted without their participation and input.

Special thanks are due to my beloved late partner Kyi Thar Hain for his continuous support and understanding until his last moment. My thanks are extended to my family members and my friends for their constant source of inspiration. Last but not least, I would like to thank my little energizer Albert (chocolate labrador).

PUBLICATIONS

RELATED TO THE THESIS :

1. T. W. W. AUNG, H. HUO, AND Y. SUI, Interactive traceability links visualization using hierarchical trace map, in 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 367-369.
2. T. W. W. AUNG, H. HUO, AND Y. SUI, A Literature Review of Automatic Traceability Link Recovery for Software Change Impact Analysis, Association for Computing Machinery, New York, NY,USA, 2020, p. 14-24.
3. T. W. W. AUNG, Y. WAN, H. HUO, AND Y. SUI, Multi-triage: A multi-task learning framework for bug triage, Journal of Systems and Software, 184 (2022),p. 111133.

TABLE OF CONTENTS

List of Publications	ix
List of Figures	xv
List of Tables	xvii
1 Introduction	1
1.1 Challenges	2
1.2 Our Approaches	5
1.3 Thesis Organization	6
2 A Literature Review of Automatic Traceability Link Recovery	9
2.1 Overview	9
2.2 Background	12
2.3 Research Method	15
2.3.1 Objectives and Research Questions	15
2.3.2 Protocol Development	16
2.3.3 Search Strategy and Data Sources	16
2.3.4 Study Selection	16
2.3.5 Quality Assessment	17
2.3.6 Data Extraction and Analysis	18
2.4 Results	18
2.4.1 Traceability Links Recovery Approaches Between Software Artifacts	18
2.4.2 Traceability Direction and Evaluation	26
2.4.3 Support Change Impact Set	27
2.4.4 Traceability Links Recovery Ways	30
2.5 Discussion	31
2.5.1 Findings of RQs	31
2.5.2 Limitations and Future Work	32

TABLE OF CONTENTS

2.6	Threats To Validity	33
2.7	Chapter Summary	34
3	Interactive Traceability Links Visualization using A Hierarchical Trace Map	35
3.1	Overview	35
3.2	Background	39
3.3	Visual Support for Traceability Links Visualization	41
3.4	Related Work	43
3.4.1	Visualization	43
3.4.2	Traceability model	45
3.5	Chapter Summary	46
4	Multi-triage : bug triage based on deep multi-task learning	47
4.1	Overview	47
4.2	Background	48
4.2.1	Developers and Issue Types Recommendation Tasks in Bug Triage	49
4.2.2	Multi-task learning	51
4.3	A motivating example	52
4.4	Multi-triage	54
4.4.1	General	54
4.4.2	Data Extraction	55
4.4.3	Contextual Data Augmenter	56
4.4.4	Multi-Triage Model	58
4.4.5	Code Representation	59
4.4.6	Task-Specific Classifiers	61
4.5	Evaluation	62
4.6	Results	65
4.6.1	RQ1: How does the multi-triage model compare to other approaches?	65
4.6.2	RQ2: Which component contributes more to the multi-triage model?	68
4.6.3	RQ3: Does increasing the size of training datasets (based on the contextual data augmentation approach) improve our model’s accuracy?	74
4.7	Threats to Validity	77
4.8	Discussion	78
4.8.1	Assessing the Significance of Our Approach	78

4.8.2	Evaluation using Time-Series Based Cross Validation	79
4.8.3	Alternative Considerations on Model Building	79
4.8.4	Applicability of Contextual Data Augmentation Approach	80
4.8.5	Lessons Learned	80
4.9	Related Work	81
4.9.1	Semi-Automatic Bug Triage	81
4.9.2	Multi-Task Learning	83
4.9.3	Other Tasks in the Bug Resolution Process	84
4.10	Chapter Summary	84
5	Case Study of Automatic Bug Triage Process Model at Software Industry	85
5.1	Overview	85
5.1.1	Contributions	86
5.2	Background	86
5.2.1	Dialog IT Issue Tracking Systems Background	87
5.3	Proposed Solution	88
5.4	Research Method	89
5.5	Design Solution	90
5.6	Results	91
5.6.1	Data analysis of historical issue reports	92
5.6.2	AI recommendation model performance	92
5.6.3	Bug triage application	94
5.6.4	Bug triage hosting components and application services	94
5.7	Chapter Summary	96
6	Conclusions and Future Work	97
6.1	Conclusions	97
6.2	Threats to Validity	98
6.3	Future Work	99
	Bibliography	101

LIST OF FIGURES

FIGURE	Page
1.1 Bug Triage Recommendation Model	3
1.2 Thesis Flow Chart	7
2.1 High-level overview of IR-based Traceability Recovery Process	9
2.2 Identify Software Change Impacts [27]	12
2.3 Overview of Traceability Literature Reviews Timeline	14
2.4 Search and selection process	17
2.5 Traceability Links Recovery Publications Trend (A rectangular box is colored according to traceability approaches. A circle symbol inside the top-right corner of the rectangle box represents the trace artifacts applied in the studies. Texts inside the rectangular box present the traceability approaches or tool names and techniques.)	19
2.6 Distribution of Publications by Traceability Direction	26
2.7 Distribution of Publications by Research Methods (on the left) and Datasets (on the right)	27
2.8 Traceability Links Recovery Ways	30
3.1 Documenting traceability relationship using textual references [126]	39
3.2 Documentation of traceability relationships using hyperlinks [126]	40
3.3 Hierarchical trace map model	41
3.4 Unfiltered view of Hierarchical Trace Map	42
3.5 Filtered view of a source artifact and two target artifacts relation	43
3.6 Filtered view of a target artifact and source artifacts relation	43
4.1 An example of an issue report and the corresponding pull request	49
4.2 Developers and issue type correlation example	50
4.3 A code snippet and corresponding AST example	52
4.4 The multi-triage framework	55

LIST OF FIGURES

4.5	An example of data extraction steps for an issue report	55
4.6	A synthetic issue report example	56
4.7	The multi-triage model	58
4.8	Time-series-based 5-fold cross-validation	64
4.9	Qualitative analysis venn diagram	67
4.10	Training time	68
4.11	Multi-triage: ablation analysis	72
4.12	Multi-triage parameter analysis	73
4.13	Multi-triage: AUC v.s. Accuracy	75
4.14	Multi-triage with Data Augmentation: AUC v.s. Accuracy	76
5.1	Issue Tracking Systems used in Sydney and Darwin Offices	87
5.2	Automatic Bug Triage Process Model	89
5.3	High-Level AI-based Bug Triage Model	91
5.4	Total no of Issue Reports Summary	92
5.5	Total no of Developers and Issue Types Summary	93
5.6	Bug Triage App Main Page	94
5.7	On-Demand Train AI Model Example	95
5.8	Retrieve Prediction Result for a New Issue Report	95
5.9	Prediction Model Results Example	96

LIST OF TABLES

TABLE	Page
2.1 Traceability Tools	25
2.2 Distribution of Publications by Change Impact Sets	28
4.1 AST nodes terms and abbreviation	53
4.2 Raw datasets information	54
4.3 Multi-triage v.s. baselines (Base1 - SVM + BOW [12], Base2 - DeepTriage [102]) Accuracy (%)	66
4.4 Qualitative analysis (bug and enhancement)	67
4.5 Single task prediction model v.s. our approach for developer predictions (pre- cision(P), recall(R), and accuracy(Acc))	69
4.6 Single task prediction model v.s. our approach for issue type predictions (precision(P), recall(R), and accuracy(Acc))	70
4.7 Unique word count for Text and AST	74
4.8 No data augmentation v.s. data augmentation (accuracy(%))	75
5.1 Research Method	90
5.2 Evaluation Results	93

