



Run Run Shaw Library

香港城市大學
City University of Hong Kong

Copyright Warning

Use of this thesis/dissertation/project is for the purpose of private study or scholarly research only. ***Users must comply with the Copyright Ordinance.***

Anyone who consults this thesis/dissertation/project is understood to recognise that its copyright rests with its author and that no part of it may be reproduced without the author's prior written consent.

CITY UNIVERSITY OF HONG KONG
香港城市大學

Enhancing Software Validation through Metamorphic
Testing: A Novel Approach
通過變形測試提升軟件驗證：一種新穎的方法

Submitted to Department of Computer Science

電腦科學系

in Partial Fulfilment of the Requirements

for the Degree of Doctor of Philosophy

哲學博士學位

by

CHAN Pak Yuen Patrick

陳栢源

APRIL 2025

二零二五年四月

Abstract

Despite machine learning technology has been applied to enhance the performance of various software systems, evaluating machine learning-based software systems is challenging, as it is difficult to distinguish correct behaviour from potentially incorrect behaviour of the machine learning-based systems. Additionally, these systems often provide a local optimum rather than a global optimum, leading to a test oracle problem. A viable approach for validating these systems is needed, and Metamorphic Testing (MT) can help alleviate the test oracle problem.

MT addresses the test oracle problem by examining the expected relationships between input-output pairs in consecutive executions of the systems. These “expected” relationships are expressed as Metamorphic Relations (MRs), which are the critical components of MT. If the output results in various software executions violate an MR, then a fault is revealed. Although MT approaches obtain encouraging results, the creation of MRs has limitations, as traditionally, MRs are created based on the inputs of experts and system users. The lack of participation from experts and users can limit the creation of effective and suitable MRs, thereby impacting the validation capacity of MT (Problem 1).

Another area of research interest is the lack of methodologically thorough performance evaluations of plagiarism detection systems, despite the serious issue of plagiarism. There is a need to raise awareness of the need to research validation approach of machine learning-based educational technologies (Problem 2).

In this thesis, we deal with Problem 1 by applying a systematic rather than a heuristic approach, to derive generic MRs of MT for validating machine learning-based software systems. These novel approaches include adapting the existing MT model into a generic MT model, deriving generic MRs of MT from the identified uncertainty, and deriving generic MRs of MT from the Metamorphic Relation

Patterns (MRPs). Additionally, we choose machine learning-based educational technologies, including plagiarism detection systems, question and answering (Q&A) websites, software defect prediction systems, and code-generation systems, as the systems under test for experiments to raise awareness and to address Problem 2.

Experiments are conducted to evaluate the performance of these proposed approaches with machine learning-based systems. Initially, we focused on adapting an existing MT model into a generic MT model. The generic MT model approach enhances the existing MT models by mapping the identified MRs of existing MT model with generic Data Mutation Operators (DMOs). The proposed generic MT model contains five generic MR types, consisting of twenty-one generic MRs, for validation. The results indicate that the generic MT model is a valuable validation tool for machine learning-based software defect prediction systems. Two machine learning-based software defect prediction systems did not violate any MR, and eighteen machine learning-based software defect prediction systems did not significantly violate one generic MR type. Additionally, results indicate that the generic MT model is an effective validation tool for software defect prediction systems with Agglomerative clustering and Density-Based Spatial Clustering of Applications with Noise machine learning algorithms.

Although enhancing existing MT approaches and adapting them into generic MT approaches can be used as a validation tool, this proposed model can only work with numeric features. This approach might not apply to other more advanced machine learning models, such as pre-trained language models. We then change the focus and experiment with the concepts of uncertainty and MRP to systematically derive generic MRs of MT.

For the concept of uncertainty, we develop a feasible approach by applying an uncertainty matrix to identify the uncertainty of machine learning-based plagiarism detection systems and derive the generic MRs of MT from the identified uncertainty for validation. Results showed that this proposed approach can effectively validate

machine learning-based plagiarism detection systems. Future studies can apply this approach to locate uncertainties to enhance systems' robustness. However, this approach might require users or experts to identify related uncertainty. Thus, we change the focus and experiment with the Metamorphic Relation Patterns (MRPs) for developing generic MRs of the MT approach that does not require users or experts.

For the MRP approach, firstly, three MRPs are identified to derive multiple generic MRs for validating machine learning-based plagiarism detection systems. The result shows that the MRP approach can systematically and effectively validate machine learning-based plagiarism detection systems. The result also demonstrates that machine learning models that build a stronger semantic relationship with words or sentences work better with MRs related to the meaning of contents. Secondly, we focused only on one pattern, the "symmetry" MRP, to derive symmetric MRs, which are also semantic-preserving MRs, to develop a more straightforward and more applicable validation approach for machine learning-based systems. Studies proved that the symmetric MRs could be a validation tool for the content quality classification on Stack Overflow Question-answering (Q&A) websites using Large Language Models (LLMs), and code-generation LLMs. In addition, the result proves that the symmetric MRs could support simulation metamorphic testing.

Furthermore, studies have discovered applications of symmetric MRs beyond validation use only. They have concluded that symmetric MRs can be utilised to explain the instability of LLMs' outputs, and as Natural Language Processing Data Augmentation (NLPDA) approach to systematically improve the machine common sense of LLMs. To the best of our knowledge, we are the first to introduce the concepts of symmetry and uncertainty to derive generic MRs of MT for validating machine learning-based systems without requiring input from users or experts. We are also the first to apply symmetric MRs to measure MT effectiveness, support simulation MT, and serve as a NLPDA to enhance the machine common sense of

machine learning-based systems. We hope the promising results will stimulate further research in these areas.

Keywords: Validation, Generic Metamorphic Testing, Uncertainty, Metamorphic Relation Pattern, Natural Language Processing Data Augmentation, Simulation Metamorphic Testing, Metamorphic Testing Effectiveness, Machine Common Sense.

CITY UNIVERSITY OF HONG KONG
Qualifying Panel and Examination Panel

Surname:	CHAN
First Name:	Pak Yuen Patrick
Degree:	Doctor of Philosophy
College/Department:	Department of Computer Science

The Qualifying Panel of the above student is composed of:

Supervisor(s):	
Prof. KEUNG Wai Jacky	Department of Computer Science City University of Hong Kong

Qualifying Panel Member(s):	
Prof. WANG Shiqi	Department of Computer Science City University of Hong Kong

Prof. SONG Linqi	Department of Computer Science City University of Hong Kong
------------------	--

This thesis has been examined and approved by the following examiners:

Prof. KEUNG Wai Jacky	Department of Computer Science City University of Hong Kong
-----------------------	--

Prof. WONG Hau San	Department of Computer Science City University of Hong Kong
--------------------	--

Prof. LEUNG Wing Ho Howard	Department of Computer Science City University of Hong Kong
----------------------------	--

Dr. HUNG King Fai Kevin	Department of Electronic Engineering and Computer Science Hong Kong Metropolitan University
-------------------------	---

Declaration

I declare that the thesis entitled “*Enhancing Software Validation through Metamorphic Testing: A Novel Approach*” represents my own work, except where due acknowledgments are made, and that it has not been previously included in a thesis, dissertation, or report submitted to this university or any other institution for a degree, diploma, or other qualifications.

CHAN Pak Yuen Patrick

April 2025

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. KEUNG Wai Jacky, who has provided me with immense support and guidance throughout my entire PhD study. His teachings and advice make this thesis possible to be completed.

I would also like to thank my PhD qualifying panel members, Dr. SONG Linqi and Dr. WANG Shiqi, not only for their precious time and insightful comments on my research reports but also for the hard questions which inspired me to widen my research from various perspectives.

I am very grateful to my dissertation examiners, Dr. WONG Hau San, Dr. LEUNG Wing Ho Howard, and Dr. HUNG King Fai Kevin, for their invaluable time spent carefully reading my thesis and for providing me with precious suggestions and constructive comments to help improve the quality of this thesis.

Then, I would like to express my special thanks to my friends and classmates, Dr. YANG Zhen and Dr. LI Hong Zong, who helped and supported me during the challenging course of the thesis. Last but not least, I owe my sincere gratitude to my family and friends, especially my brother Cheho, who has always been supportive.

List of Publications

Published Papers

1. **P. Y. P. Chan**, J. Keung, and Z. Yang, “Effectiveness of Symmetric Metamorphic Relations on Validating the Stability of code-generation LLM,” *Journal of Systems and Software*, vol. 222, article no. 112330, Apr. 2025.
2. **P. Y. P. Chan** and J. Keung, “Validating pretrained language models for content quality classification with semantic-preserving metamorphic relations,” *Natural Language Processing Journal*, vol. 9, article no. 100114, Dec. 2024.
3. **P. Y. P. Chan**, and J. Keung, “Validating Unsupervised Machine Learning Techniques for Software Defect Prediction with Generic Metamorphic Testing,” *IEEE Access*, vol. 12, p. 165155 – 165172, Nov. 2024.
4. **P. Y. P. Chan** and J. Keung, “A Symmetric Metamorphic Relations Approach Supporting LLM for Education Technology,” International Symposium on Educational Technology, Macao SAR, China, Jul. 29 – Aug. 1, 2024.
5. **P. Y. P. Chan**, J. Keung, and Z. Yang, “Uncertainty-Based Metamorphic Testing for Validating Plagiarism Detection Systems,” *Communications in Computer and Information Science* Vol. 1974, Springer, 2024, pp. 299-314.
6. **P. Y. P. Chan**, J. Keung and Z. Yang, “Validating Plagiarism Detection Systems with Metamorphic Testing,” *International Symposium on Educational Technology (ISET)*, Ho Man Tin, Hong Kong, 2023, pp. 132-137.
7. **P. Y. P. Chan**, J. Keung, and Z. Yang, “Identifying Inconsistent Software Defect Predictions with Symmetric Metamorphic Relation Patterns,” *Journal of Systems and Software*. (Accepted Date: April, 2025)

Under Review Papers

1. **P. Y. P. Chan** and J. Keung, “Identifying the cause of performance issues of PLM for Educational Technology,” *International Symposium on Educational Technology (ISET)*, Bangkok, Thailand, Jul. 22 – 25, 2025. (Submitted Date: Feb, 2025)

Contents

ABSTRACT.....	I
DECLARATION.....	III
ACKNOWLEDGEMENTS.....	IV
LIST OF PUBLICATIONS.....	V
CHAPTER 1. INTRODUCTION	15
1.1 BACKGROUND	15
1.2 MOTIVATION	17
1.2.1 Problem 1. Difficulty in identifying Metamorphic Relations of Metamorphic Testing	17
1.2.2 Problem 2. Lack of awareness of the need to research validation approach of machine learning-based educational systems.....	18
1.3 CONTRIBUTION OF THE THESIS	19
1.4 ORGANIZATION OF THE THESIS.....	19
CHAPTER 2. LITERATURE REVIEW	21
2.1 RISKS IN APPLYING MACHINE LEARNING-BASED SOFTWARE SYSTEM.....	21
2.1.2 Validating the performance of machine learning-based systems and Test oracle	22
2.2 METAMORPHIC TESTING	23
2.2.1 Basic Concept.....	24
2.2.2 Application of Metamorphic Testing	24
2.2.3 Identifying Metamorphic Relations of Metamorphic Testing.....	25
2.3 CHAPTER SUMMARY	26
CHAPTER 3. VALIDATING MACHINE LEARNING-BASED SYSTEMS WITH GENERIC METAMORPHIC TESTING	27
3.1 INTRODUCTION	28
3.2 BACKGROUND	29
3.2.1 Software Defect Prediction.....	29
3.2.2 Machine Learning and Test Oracle	30
3.3 MOTIVATION	31
3.4 PROPOSED APPROACH.....	31
3.4.1 Generic Metamorphic Testing model.....	32

3.5 EXPERIMENT WITH GENERIC METAMORPHIC TESTING MODEL ON VALIDATING MACHINE LEARNING-BASED SOFTWARE DEFECT DETECTION SYSTEMS.....	36
3.5.1 <i>Research Questions</i>	36
3.5.2 <i>Subject Machine learning algorithm</i>	37
3.5.2.1 Unsupervised Software Defect Prediction models (USDPs).....	37
3.5.2.2 Supervised Software Defect Prediction models (SSDPs)	41
3.5.3 <i>Subject Dataset</i>	42
3.5.4 <i>Environment</i>	44
3.5.5 <i>Measurement</i>	44
3.5.6 <i>Implementation</i>	46
3.5.7. <i>Result and Discussion</i>	47
3.5.7.1 RQ1: Could generic MT model validate SDPs?	48
3.5.7.1.1 Zero violation with all MRs	48
3.5.7.1.2. Zero violation with MR2.....	49
3.5.7.1.3. Violated all MRs	50
3.5.7.1.4. Violated MR1, MR3, MR4 and MR5.....	51
3.5.7.2 RQ2: What are the underlying causes of violations?	56
3.5.7.2.1 Caused by characteristics of data	56
3.5.7.2.2 Caused by configurations of algorithms.....	58
3.5.7.2.3 Caused by characteristics of algorithms	59
3.5.7.3 RQ3: What are the differences in applying generic MT model to validate USDP and SSDP?	61
3.5.7.3.1 MR2	62
3.5.7.3.2 Correlation between prediction accuracies and inconsistencies	62
3.5.7.4 Further Analysis.....	63
3.5.8 <i>Threats to Validity</i>	64
3.5.8.1 Internal validity.....	64
3.5.8.2 External validity.....	64
3.5.8.3 Construct validity.....	65
3.5.9 <i>Limitations</i>	65
3.6 CHAPTER SUMMARY	66
CHAPTER 4. UNCERTAINTY-BASED METAMORPHIC TESTING ON VALIDATION OF MACHINE LEARNING-BASED SYSTEMS.....	68
4.1 INTRODUCTION	69
4.2 BACKGROUND	70
4.2.1 <i>Plagiarism Detection</i>	70
4.2.2 <i>Uncertainty</i>	71
4.2.3 <i>Metamorphic Testing (MT)</i>	72

4.3 MOTIVATION	72
4.4 PROPOSED APPROACH.....	72
4.4.1 <i>Uncertainty-Based Metamorphic Testing</i>	72
4.5 EXPERIMENTAL WITH UNCERTAINTY-BASED METAMORPHIC TESTING ON VALIDATION OF MACHINE LEARNING-BASED PLAGIARISM DETECTION SYSTEMS	74
4.5.1 <i>Research Questions</i>	74
4.5.2 <i>Experiment scenario</i>	74
4.5.3 <i>Subject systems and technologies under test (SUT)</i>	75
4.5.3.1 Plagiarism detection system (PDS).....	75
4.5.3.2 Paraphrasing tools.....	75
4.5.3.3 Explainable Artificial Intelligence (XAI)	76
4.5.4 <i>Configuration</i>	76
4.5.5 <i>Subject Data</i>	77
4.5.6 <i>Measurement</i>	78
4.5.7 <i>Experiment implementation</i>	78
4.5.8 <i>Experimental Results and Discussion</i>	79
4.5.8.1 Zero violation	79
4.5.8.2 Violation with MR1 and BERT_COS.....	81
4.5.8.3 Violation with MR1 and NN_TFIDF.....	81
4.5.8.4 Violation with MR1 and Neu_SBERT	83
4.5.8.5 Further Analysis.....	84
4.5.9 <i>Validity</i>	85
4.5.9.1 Internal validity.....	85
4.5.9.2 External validity.....	85
4.5.10 <i>Limitations</i>	86
4.6 CHAPTER SUMMARY	86
CHAPTER 5. METAPHORIC RELATION PATTERNS OF METAMORPHIC TESTING FOR VALIDATING MACHINE LEARNING-BASED SYSTEMS.....	88
5.1 INTRODUCTION	89
5.2 BACKGROUND	89
5.2.1 <i>Plagiarism</i>	90
5.2.2 <i>Similarity</i>	90
5.2.3 <i>Plagiarism Detection with machine learning</i>	90
5.3 MOTIVATION	91
5.4 PROPOSED APPROACH.....	92
5.4.1 <i>Generic Metamorphic Relations Pattern (MRP)</i>	92

5.5 EXPERIMENTAL WITH METAMORPHIC RELATIONS PATTERN OF METAMORPHIC TESTING MODEL ON	
VALIDATING MACHINE LEARNING-BASED PLAGIARISM DETECTION SYSTEMS	93
5.5.1 Research Questions	93
5.5.2 Configuration	93
5.5.3 Subject Dataset.....	94
5.5.4 System Under Test (SUT).....	94
5.5.5 Query.....	95
5.5.6 Implementation	97
5.5.7 Experimental Results and Discussion	98
5.5.7.1 MR1 - Shuffling.....	98
5.5.7.2 MR2 – Input equivalence.....	100
5.5.7.3 MR3 – Partition difference	101
5.5.8 Threats to validity.....	101
5.5.8.1 Internal validity.....	101
5.5.8.2 External validity.....	102
5.5.9 Limitations.....	102
5.6 CHAPTER SUMMARY	102
CHAPTER 6. SYMMETRIC METAPHORIC RELATIONS OF METAMORPHIC TESTING FOR	
VALIDATING MACHINE LEARNING-BASED SYSTEMS.....	104
6.1 INTRODUCTION	105
6.2 BACKGROUND	107
6.3 MOTIVATION	108
6.4 PROPOSED APPROACH.....	108
6.4.1 Symmetric Metaphoric Relations of Metamorphic Testing.....	108
6.5 EXPERIMENTAL DESIGN	109
6.5.1 Experiment with Symmetric Metaphoric Relations on validating the code generation LLM.....	109
6.5.1.1 Background.....	110
6.5.1.1.1 Code Generation	111
6.5.1.1.2 Large language model (LLM) for code generation and Metamorphic Testing	111
6.5.1.2 Research Question	113
6.5.1.3 Proposed Metamorphic Relations (MRs).....	113
6.5.1.4 Experimental Settings.....	115
6.5.1.4.1 Test Scenario.....	115
6.5.1.4.2 Subject Under Test (SUTs)	116
6.5.1.4.3 Subject Dataset	120
6.5.1.4.4 Environment.....	121
6.5.1.4.5 Measurement.....	121

6.5.1.4.6 Implementation	127
6.5.1.4.7 Results and discussion	128
6.5.1.4.8 Threats to Validity	144
6.5.1.4.9 Limitations	146
6.5.1.5 Conclusion	146
6.5.2 <i>Experiment with Symmetric Metaphoric Relations on validating the content quality classification of Pre-trained Language Models</i>	148
6.5.2.1 Background.....	149
6.5.2.1.1 Question-answering (Q&A) websites	149
6.5.2.1.2 Apply Machine Learning Approach to Q&A	150
6.5.2.2 Research Question	151
6.5.2.3 Proposed Metamorphic Relations (MRs).....	151
6.5.2.3.1 MRG1 – Duplication.....	152
6.5.2.3.2 MRG2 – Abbreviation replacement	152
6.5.2.3.3 MRG3 – Uppercase conversion	153
6.5.2.4 Experimental Setting.....	154
6.5.2.4.1 Test scenario	154
6.5.2.4.2 Subject pretrained language model	154
6.5.2.4.3 Subject Dataset	155
6.5.2.4.4 Environment.....	156
6.5.2.4.5 Measurement.....	156
6.5.2.4.6 Implementation	157
6.5.2.4.7 Experimental results and discussion	159
6.5.2.4.8 Threats to Validity	165
6.5.2.4.9 Limitations	166
6.5.2.5 Conclusion	167
6.5.3 <i>Experiment with Symmetric Metaphoric Relations on enhancing Machine common sense of LLM</i>	168
6.5.3.1 Background.....	169
6.5.3.1.1 Question-Answering (Q&A) websites	169
6.5.3.1.2 Large language model (LLM).....	169
6.5.3.1.3 Machine common sense and natural language processing data augmentation	170
6.5.3.2 Research Question	170
6.5.3.3 Proposed metamorphic relation (MR).....	170
6.5.3.4 Experimental Setting.....	171
6.5.3.4.1 Test Scenario.....	171
6.5.3.4.2 Subject large language model	171
6.5.3.4.3 Subject Dataset	172

6.5.3.4.4 Setup	173
6.5.3.4.5 Measurements	173
6.5.3.4.6 Implementation	174
6.5.3.4.7 Experimental results and discussion	175
6.5.3.4.8 Threats to Validity	178
6.5.3.4.9 Limitation	179
6.5.3.5 Conclusion	180
6.6 CHAPTER SUMMARY	180
CHAPTER 7. CONCLUSION AND FUTURE WORK.....	183
7.1 SUMMARY	183
7.2 CONTRIBUTIONS	185
7.3 FUTURE WORKS.....	186
7.3.1 Further explore and investigate Symmetry MRP	186
7.3.2 Explore and investigate other generic MT approaches	186
7.3.3 Explore and investigate enhancing machine common sense.....	187
7.3.4 Better understanding the machine learning-based systems	187
7.3.5 Explore and investigate other approaches to enhance the performance of machine learning-based systems.....	188
7.3.6 Explore and investigate other approaches to improve proposed approaches	189
BIBLIOGRAPHY	190
APPENDIX A.....	216

List of Tables

Table 3.1: Mapping table of MRs and DMOs.....	32
Table 3.2: List of subject machine learning algorithms	41
Table 3.3: Information of subject datasets.	43
Table 3.4: Violation Pattern of Relabelling after apply MRs.....	46
Table 3.5: Multiple paired t-tests with significance level at 0.01 for mean of difference in average RP values of MR2 between Agglo and Affinity which were significantly different.	49
Table 3.6: Average RP, VR, ACC and MCC values of SDPs with all MRs and all test datasets.	50
Table 3.7: Average RP values of MR2 of SDPs with all test datasets.	50
Table 3.8: Average RP of Optics with violated test datasets.	51
Table 3.9: Multiple paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and MiniKM.....	54
Table 3.10: Multiple paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and BaygGM(spherical).	55
Table 3.11: The statistic of the most different feature “HALSTEAD_DIFFICULTY” between CM1, PC1 and MW1.	57
Table 3.12: Correlation analysis based on average values of all SDPs.....	62
Table 4.1: Alwidian et al. [129]’s uncertainty matrix	73
Table 4.2: List of online free paraphrasing tools	76
Table 4.3: Details of subject data.....	77
Table 4.4: Illustration of violating MR1	78
Table 4.5: Violation counts in the experiment	79
Table 4.6: Matrix of similarity scores of BERT_COS with CS	80
Table 4.7: Matrix of similarity scores of NN_TFIDF with CS	80
Table 4.8: Matrix of similarity scores of Neu_SBERT with CS	80
Table 5.1: Contents of five source queries.....	96
Table 5.2: Contents of follow-up queries 1.1, 1.2, 2.1 and 2.2	96
Table 5.3: Contents of Follow-up Queries 3.1 and 4.1	96
Table 5.4: Contents of Follow-up Queries 5.1, 5.2 and 5.3	97
Table 5.5: MR1 – Paired T-Test results.....	99
Table 5.6: MR2 – Paired T-Test Results.....	100
Table 5.7: MR3 – Paired T-Test Results.....	100

Table 6.1: Descriptions of MRs.	115
Table 6.2: Illustration of effective measurements.	125
Table 6.3: Illustration of the similarity and effective calculations.	125
Table 6.4: VR values of all LLMs with all MRs in seen project.....	130
Table 6.5: VR values of all LLMs with all MRs in unseen project.....	130
Table 6.6: TS, FS, TV, and FV values of all LLMs with all MRs in unseen project.	133
Table 6.7: GSR values of all LLMs with all MRs in unseen project.	134
Table 6.8: TS, FS, TV, and FV values of all LLMs with all MRs in seen project.	135
Table 6.9: GSR values of all LLMs with all MRs in seen project.	136
Table 6.10: ESR values of all LLMs with “Source” and all MRs in seen project.....	136
Table 6.11: pass@1 values of all LLMs with “Source” and all MRs in unseen project	137
Table 6.12: The percentage of total cases that have consistent results between “Source” outputs and ground truth.....	137
Table 6.13: Table of the proposed semantic-preserving MRs.....	153
Table 6.14: RR values of four PLMs with all MRs.	159
Table 6.15: ACC values of four PLMs with all MRs and Source.	160
Table 6.16: MCC values of four PLMs with all MRs and Source.	160
Table 6.17: Four extra MRs related to uppercase conversion.	161
Table 6.18: RR values of GPT1 with extra MRs related to uppercase conversion.	161
Table 6.19: The CSAT, ACC and MCC results of the experiment.	177

List of Figures

Figure 3.1: Illustration of MR1 data transformation will not affect the clustering results.	33
Figure 3.2: (a) Illustration of MR2 shrinking in proportion will not affect the clustering results. (b) Illustration of MR2 expanding in proportion will not affect the clustering results.	34
Figure 3.3: (a) Illustration of MR3 increases density with instances that are shrinking in proportion will not affect the clustering results. (b) Illustration of MR3 increases density with instances that expanding in proportion will not affect the clustering results.	34
Figure 3.4: Illustration of MR5 (a)MR5.1 (b)MR5.5 (c)MR5.6 (d)MR5.7 (e)Outliers will not affect the clustering results due to the distances	35
Figure 3.5: Illustration of the experiment	47
Figure 3.6: Box and whisker diagram of average RP values of all datasets with (a)all SDPs (b)Affinity (c)Meanshift (d)Birch (e)KNN (f)RF (g)Kmean(full) (h)MiniKM (i)GMM(spherical) (j)BayGM(spherical).....	53
Figure 4.1: Illustration of MR1: Paraphrasing object.....	73
Figure 4.2: Implementation of experiment.....	79
Figure 4.3: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)std_answer-cs, (b)gpt-org-cs, (c)gpt-p1-cs.	82
Figure 4.4: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)cla-org-law1, (b)gpt-org-law1, (c)gpt-p1-law1.....	82
Figure 4.5: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)gpt-p2-law2, (b)cla-p3-law2, (c)gpt-p5-law2.	83
Figure 5.1: The implementation process of the experiment	97
Figure 6.1: Illustration of Step 1.	128
Figure 6.2: Illustration of Step 2 to Step 5.	128
Figure 6.3: Illustration of the experiment (Step 1 to 5).....	158
Figure 6.4: Illustration of dataset creations.	173
Figure 6.5: Illustration of Step 1 to Step 3.	175
Figure 6.6: Illustration of Step 4 to Step 5.	175

Chapter 1. Introduction

In software engineering, evaluating software systems is crucial as their quality is directly linked to user trust. With the increasing application of machine learning algorithms to enhance software performance, evaluating machine learning-based systems has become more challenging due to the test oracle problem. Therefore, an effective validation technique is necessary to ensure the accuracy of machine learning-based software systems and maintain user confidence [1]. Given the widespread application of machine learning algorithms across various domains, there is a need for research on effective validation tools. Generic metamorphic testing has been identified as a potential validation tool; however, research in this area, especially concerning the evaluation of machine learning-based educational systems, is limited. This thesis is focused on the study of enhancing software validation through various novel approaches to metamorphic testing. In this chapter, we will first introduce the background of research on metamorphic testing-based software validation, and then outline the motivation and contribution of this thesis.

1.1 Background

Machine learning (ML) algorithms have been applied to different areas, such as plagiarism detection [2], software defect detection [3-6], content quality classification [7], Code generation [8,9], etc. Despite the advantages of applying ML,

evaluating ML-based software systems is challenging. Although, there are several evaluation approaches for ML-based code generation systems [9-18], those evaluation approaches focus mainly on the performance of code generation rather than validating the ML-based code generation systems themselves. Thus, the test oracle problem remains because it is difficult to distinguish correct behaviour from potentially incorrect behaviour and ML-based systems often provide a local optimum rather than a global optimum [19-24].

We see the potential of Metamorphic testing (MT) can be a valuable validation tool for ML-based software systems because MT has been used for oracle problem in different domains, such as bioinformatic software [25], machine learning classification [26-29], cryptographic software [30], online search engines [31]. MT can also be applied in validating autonomous vehicles systems [32,33], fake news detection systems [34], automatic image classification systems [35], and multiple linear regression systems [36]. It is also plausible to define metamorphic relation patterns (MRPs) for deriving multiple MRs [31]. Furthermore, it is plausible to improve the capacity of MT by introducing simulation testing concept [37] and by introducing effectiveness measurement of MT [38].

Metamorphic Testing (MT) was introduced by [39] as a novel approach to alleviate the problem of test oracles by verifying the relations between the inputs and outputs across multiple test executions of the programs or systems under examination. These relationships, known as Metamorphic Relations (MRs), embody the essential attributes of the target function or algorithm concerning the multitude of inputs and their anticipated outputs [39, 40]. The methodology of MT begins with the creation of specific program inputs known as “Source” inputs, which serve as the foundational test cases. Using these “Source” test cases, MRs are applied to generate new inputs, referred to as “Follow-up” test cases. In contrast to traditional testing methods, MT thoroughly verifies both the “Source” and “Follow-up” test cases, as well as their corresponding outputs, against the defined MRs [39, 40].

This thesis aims at researching systematic approaches to derive generic MRs of MT for evaluating ML-based software systems. The first phase of studies is enhancing existing MT approaches and adapting them into generic MT approaches for validation, followed by a second phase that aims to experiment with the concepts of uncertainty and symmetry in systematically deriving generic MRs of MT.

1.2 Motivation

Despite the encouraging results obtained by existing MT approaches in evaluating ML-based software systems, it requires the inputs of experts and users for the creation of MRs. Another problem is the lack of research interest in assessment of ML-based educational technologies. We describe these two problems in detail below.

1.2.1 Problem 1. Difficulty in identifying Metamorphic Relations of Metamorphic Testing

Evaluating the performance of ML-based software systems is a test oracle problem because the systems can only predict at best with the given training data and can only find the local optimum rather than the global optimum, and MT is a commonly used approach to address the test oracle problem [19-21, 23, 24]. However, most of the MRs of MT are derived from the inputs of users or experts, and if in a situation in which software modules have limited historical data and limited inputs of users or experts, MRs could be challenging to determine [19, 23, 41]. It is crucial to have generic MRs for evaluation, as experts or users might not be available, and there might be limited knowledge of the trained ML models or other newly created systems. To address this problem, this thesis aims to experiment with different novel approaches to derive generic MRs of MT for validating different ML-based systems.

1.2.2 Problem 2. Lack of awareness of the need to research validation approach of machine learning-based educational systems

ML technology has been applied for enhancing the performance of educational systems, including the performance of detection of text documents such as fraudulent reviews, spam filtering, tag extraction and plagiarism [42-45]. A systematic review study has found that plagiarism detection was improved by “Better semantic text analysis methods” and “Application of machine learning” [2]. To apply ML techniques to detecting text documents, the ML model typically works with Natural Language Processing (NLP). NLP is one of the computation techniques to capture the semantics of natural language texts and uses different knowledge representations for specific purposes [46-48]. ML models can be based on representations of NLP to make predictions. Additionally, ML has been applied to maintain the quality of Question and Answer (Q&A) websites by applying pre-trained language models (PLMs) to either replying the users’ questions or classifying the content quality of the sites automatically [7, 49, 50].

Despite advanced technologies have been applied in ML-based Plagiarism Detection Systems (PDS) to enhance performance [42-45], other advanced technologies have also made detection more complex [51, 52]. Thus, validating the PDSs themselves is essential because plagiarism is a severe issue in academia and incorrect detections will cause disputes [51] and reduce trust in using the systems [1]. However, ML models are difficult to assess [53], especially since there is a lack of research interest in ML-based educational technologies, such as PDSs. Researchers [2, 54, 55] found a research gap in the lack of methodologically thorough performance evaluations of PDSs. There is a need to raise awareness of the need to research the validation approach of ML-based educational technologies. To address this problem, this thesis uses ML-based educational technologies as systems under test in the experiments, to bring awareness of research focus.

1.3 Contribution of the Thesis

The proposed approaches of this thesis aim to enhance the validation capacity of MT with ML-based systems. The main contributions of this thesis include:

- Proposed novel approaches for validating ML-based systems, specifically in under-researched areas like validating ML-based PDSs and educational technologies, without requiring input from users or experts.
- Demonstrated that generic metamorphic relations (MRs) of metamorphic testing (MT) can be extended to validate various domains, using different languages and ML algorithms, including pre-trained Large Language Models (LLMs).
- Developed effective methods for deriving generic MRs accessible to researchers and practitioners, regardless of their prior knowledge.
- Discovered that symmetric MRs, derived from symmetry metamorphic pattern (MRP), can serve as a general validation tool for different ML-based systems. Furthermore, symmetric MRs can be used as a formal Natural Language Processing Data Augmentation (NLPDA) approach to provide trustworthy training or testing datasets, to create non-violation regions to support simulation MT, and to explain the instability of ML-based systems' outputs.

1.4 Organization of the Thesis

The rest of this thesis is organised as follows. In Chapter 2, we first review the difficulties in validating ML-based software systems and briefly describe MT and its applications in the context of software engineering.

In Chapter 3, we investigate the possibility of using generic MRs of MT to validate ML-based systems by introducing an adapted generic MT model to validate ML-based SDPs. Without the input of users and experts, we adapt an existing MT model

into a generic MT model, which includes five types of MR mapping with data mutation operators (DMOs). Experiment results show that the model is a valuable validation tool for ML-based SDPs, even with limited knowledge of validating targets.

We develop another feasible approach in Chapter 4 by applying an uncertainty matrix to identify the uncertainty of the ML-based PDSs and derive generic MRs from the identified uncertainty for validation. We experimented with three PDSs in a classroom scenario. Results show that the proposed approach can effectively validate PDSs and enhance the systems' robustness.

In Chapter 5, we derive three generic MRs of MT based on MRPs to validate PDSs using different ML algorithms. Experiment results show that generic MRs can systematically and effectively validate and compare PDSs. We also concluded that ML algorithms that build a stronger semantic relationship with words or sentences work better with MRs related to the meaning of contents.

Chapter 6 focuses on the symmetry MRP to derive symmetric MRs for validating ML-based systems, including code-generation systems, content quality classification systems, and software defect prediction systems. Experiment results show symmetric MRs can serve as a validation tool for different ML-based systems, and these relations can also be applied to enhance the machine common sense of the systems, to support simulation MT, and to explain the instability of ML-based systems' outputs.

We conclude the thesis in Chapter 7, which includes the findings drawn from the research, contributions, and suggestions for future work.

Chapter 2. Literature Review

Machine learning technologies have been applied to various domains for enhancing performances; however, performance evaluation of machine learning (ML) models is a test oracle problem since ML models deliver local optima rather than global optima [19, 23, 24, 37]. In this chapter, we first discuss the risks of applying machine learning (ML) to different areas; then, we discuss metamorphic testing (MT) in the context of software engineering, which is used to validate machine learning-based (ML-based) systems in this thesis.

2.1 Risks in applying Machine learning-based software system

Various areas, such as software engineering and education, have used ML technologies to enhance performance [4, 42-45, 56, 57]. Pre-trained Large Language models (LLMs) have been applied to respond to many questions and maintain the quality of the large volume of content on different Question-and-Answer (Q&A) platforms [7, 49, 50, 58-63]. With the advancements in ML technologies, researchers and software developers have shifted their focus towards leveraging ML technologies in different software engineering tasks, and advanced code generations have been developed because of applying LLMs, which have shown remarkable effectiveness in solving complex tasks [8, 9, 12, 15, 16, 64, 65]. Another software

engineering task that utilised ML technologies to enhance performance is Software Defect Prediction (SDP). SDP helps to identify defect tendencies in software modules, and there are many ML-based approaches to predicting software defects [3-6, 56, 66, 67].

However, advanced ML technologies also led to other new risks and problems. Advanced technologies, such as Artificial Intelligence Generative Content (AIGC), have made detecting plagiarism more difficult [51, 52], especially since there is still no system that can detect plagiarism 100% correctly [51]. There was a lack of performance evaluation of Plagiarism Detection Systems (PDSs) [54, 55]. Although there are many advantages of using pre-trained Large Language Models (LLMs), companies that use third-party generative Artificial Intelligence (AI) tools or LLMs may raise safety and security concerns and risk exposing corporate secrets [68, 69]. LLMs have also been widely utilised to uphold content quality on various Q&A websites, including medical sites and Stack Overflow platforms [7, 49, 50, 59, 60, 70]. Although the performance of LLMs is generally satisfactory, human reviews and evaluations remain necessary to ensure the safe utilisation of LLMs [49, 50, 53]. In addition, many LLMs' performances showed that they do not possess deep semantic comprehension or human-level common sense [71, 72]. Thus, there is still a gap in LLM's ability to reach robust human-level common sense reasoning [71]. Another risk is that even though the risk of applying ML is a genuine concern, there was a lack of research interest in evaluating the performance of ML-based systems, especially ML-based PDSs [2, 54, 55]. Thus, the literature showed a need to research assessment approaches for ML-based systems.

2.1.2 Validating the performance of machine learning-based systems and Test oracle

Prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) [73] are two of the most used metrics to evaluate the performance of ML-based systems under

the situation that the labelled datasets are available, which means the ground truths are available for evaluation. In addition, different metrics and approaches are proposed to evaluate ML-based code generation systems (CGs), such as pre-defined code testing datasets and execution successful rates of the generated codes [9-16, 18, 74]. However, most evaluation approaches focus on the performance of ML-based systems rather than validating the systems themselves, and without validating the systems, the performance of the systems can be contaminated. Validating ML-based systems presents a challenge known as the test oracle problem, which makes it challenging to differentiate between correct behaviour and potentially incorrect behaviour [19, 23, 24, 37]. In addition, if the ML-based systems are unsupervised, which are trained with unlabelled datasets (datasets without ground truth), it is difficult to evaluate them as there is no prior knowledge of the unlabelled datasets, making this an untestable situation, also referred to as the test oracle problem [5, 19-24, 27]. Consequently, those evaluation approaches mentioned, especially ACC and MCC, which require ground truth to measure with, cannot be used for the test oracle problem. A systematic and practical approach is needed to address the test oracle issue, and Metamorphic testing (MT) can be employed to mitigate this problem [19, 22-24].

2.2 Metamorphic Testing

Metamorphic Testing (MT) was introduced by [39] as a novel approach to alleviate the problem of test oracles by verifying the relations between the inputs and outputs across multiple test executions of the programs or systems under examination. These relationships, known as Metamorphic Relations (MRs), embody the essential attributes of the target function or algorithm concerning the multitude of inputs and their anticipated outputs [39, 40]. The methodology of MT involves the initial generation of specific program inputs, referred to as “Source” inputs, to serve as the foundational test cases. Subsequently, leveraging these “Source” test cases, MRs are

employed to generate new inputs, thus constituting the “Follow-up” test cases. Unlike the conventional testing methodology, MT comprehensively verifies the “Source” and “Follow-up” test cases and their corresponding outputs against the specified MR [39, 40].

2.2.1 Basic Concept

MT has been used to validate the “expected” relationships between inputs and outputs of program executions. These “expected” relationships are expressed as MRs, and they are the core element of MT, and if the output results in various software executions violate an MR, then a fault is revealed [19, 21, 24, 31]. So, by examining these expected relationships between inputs and outputs, which are MRs, we can validate the program under testing and avoid the test oracle problem. As machine learning is composed of programs and datasets, if the machine learning model is insensitive to data transformation, then any violation of MR can be interpreted as a fault in the program.

The idea of MR can be illustrated by a mathematical property of the *sine* function, which is “ $\sin(x) = \sin(\pi - x)$ ”. For example, if the corresponding MR has two inputs, x_1 and x_2 , that satisfy “ $x_1 + x_2 = \pi$ ”, then two function outputs should be equal, i.e., $\sin(x_1) = \sin(x_2)$. Based on MR, x_2 is constructed based on x_1 and $\sin(x_1)$. Therefore, x_1 is the source input, and x_2 is the follow-up input. Such MR can be used to test the program as if the outputs of $\sin(x_1)$ and $\sin(x_2)$ are different, then the implementation of function “ $\sin()$ ” must have faults as it violates the above MR [23].

2.2.2 Application of Metamorphic Testing

MT has been used for the test oracle problem in different domains, such as bioinformatics software [25], machine learning classification [26-29], cryptographic software [30], and online search engines [31]. MT can also be applied in validating autonomous vehicle systems [32, 33], fake news detection systems [34], automatic

image classification systems [35], multiple linear regression systems [36], software testing in System of Systems [75], identify incorrect compilations in Deep Learning compilers [20] and validate the test case generation systems [23].

It is also plausible to define metamorphic relation patterns (MRPs) for deriving multiple MRs [31]. [24]’s study further the concept of metamorphic relations and symmetry to define metamorphic relation patterns (MRPs) that can derive multiple MRs. Furthermore, it is plausible to improve the capacity of MT by introducing a simulation testing concept to enable testers or researchers to determine whether the test case is violated or not by the location of the test case [37] and by introducing effectiveness measurement of MT, which refers to measuring how well the evaluation results from MT can reflect the actual performance or target system [38].

2.2.3 Identifying Metamorphic Relations of Metamorphic Testing

We see the potential of MT in validating ML-based systems, and the key to MT application is the identification of MRs. However, in a situation in which software modules have limited historical data and limited experts’ or users’ inputs, MRs could be challenging to determine [19, 23, 41]; thus, a generic approach can be applied to rescue this situation. [24]’s study proposed the concepts of symmetry to derive multiple concrete metamorphic relations that can be applied in different application domains. [31]’s study presented a list of generic MRPs to assist in identifying MRs when testing query-based systems. [75]’s system of systems testing study showed that abstract MRP and the concept of symmetry can generate generic MRs. Other studies proved that the μ MT model, which used Data mutation operators (DMOs) to construct generic MRs, is feasible [41]. Therefore, identifying MRs can also be achieved by providing a set of DMOs and mapping rules of μ MT [76]. Thus, this thesis aims to research different novel and systematic approaches to derive generic MRs of MT to validate ML-based systems.

2.3 Chapter Summary

In this chapter, we describe the risks of applying ML; then, we discuss MT as a potential validation tool of ML-based systems and the problem related to identifying MRs of MT. Our objective is to explore various novel and systematic approaches to derive generic MRs of MT to effectively validate ML-based systems. This thesis aims to demonstrate the practicality and robustness of different approaches in deriving generic MRs of MT as a validation tool through several comprehensive scaled experiments. In the next chapter, we conduct an experiment to assess the generic MT as a validation tool for ML-based systems.

Chapter 3. Validating Machine learning-based Systems with Generic Metamorphic Testing

In this chapter, we present a novel approach that adapts an existing metamorphic testing (MT) model into a generic MT model and utilises generic MT to effectively validate unsupervised software defect prediction models. It is imperative to address the challenges associated with insufficient involvement from experts and users, which can impede the development of effective and appropriate metamorphic relations (MRs) and ultimately affect the validation capacity of MT.

Our method identifies five categories of MRs of an existing MT, further divided into twenty-one individual generic MRs, all formulated through generic Data Mutation Operators (DMOs). This framework enables us to generate follow-up datasets from the source datasets, training respective software defect prediction models (SDPs). By comparing predictions between the “source” and “follow-up” SDPs and identifying inconsistencies, we can assess the SDPs’ sensitivity to generic MRs as a form of validation.

The rest of this chapter is organised as follows. Section 3.1 introduces this experiment. Section 3.2 describes the underlying concepts of software defect prediction using the machine learning approach and MT. Section 3.3 describes the

motivation behind this experiment. Section 3.4 describes our proposed generic MT model. Section 3.5 describes the experimental setup to determine the effectiveness of the proposed model in validating SDPs and discusses the results and the threats to the validity of this study. Section 3.6 concludes this chapter.

3.1 Introduction

As software complexity grows, defect prediction has gained more research interest in software engineering, particularly through machine learning (ML) technologies [3, 4, 56, 57]. Supervised learning models with labelled software defect datasets, which are Supervised Software Defect Prediction models (SSDPs), are widely used [56]. However, creating labelled software defect datasets is challenging when the software projects have limited historical software data [56]. SSDPs cannot be applied to datasets that do not have labels. Thus, Unsupervised Software Defect Prediction models (USDPs) become the solution as USDPs do not need labelled datasets [56, 77]. However, USDPs pose validation challenges due to the lack of prior knowledge about unlabelled datasets, leading to the test oracle [19].

Despite the testing challenge, MT is a widely used technique to alleviate the test oracle problem [19-24]. Instead of verifying the outputs of software programs, MT validates the expected relationships between inputs and outputs of multiple executions of the software and these relationships are called MRs. If any MR is violated after the execution of software, it is interpreted as a faulty implementation. Through this approach, we avoid the test oracle problem. However, identifying MRs is challenging because these expected relationships were mainly created by experts or users [19, 23, 41].

To alleviate the challenge of identifying MR, we proposed a generic and adaptive MT model for validating SDPs. Different from traditional models, this model is generic and based on two proven models with no expert or user input or prior knowledge needed. Five generic types of MRs are identified from the

“METamorphic Testing approach to assessing and validating unsupervised machine Learning” model (METTLE) [19]. They are mapped with DMOs of the μ MT model [41, 76] and proposed DMOs to establish the generic MT model with twenty-one generic MRs in this study.

We evaluate this generic MT model by conducting an experiment with twenty SDPs using fourteen different ML algorithms on twenty publicly available high-dimensional datasets, in order to achieve our research objective, which is to demonstrate the feasibility of the proposed generic MT approach for validating USDPs using clustering algorithms and SSDPs. We applied the MRs to generate follow-up datasets to train follow-up SDPs for testing. The predictions of follow-up SDPs would be compared with predictions of SDPs trained by the source datasets. If there is no violation, which means no inconsistent prediction between them, the SDP is insensitive to the related MR and that MR could be used for validation. It is because if SDP is insensitive to data transformation, then any violation of MR can be interpreted as a fault in the program. The results show that the proposed model can effectively validate SDPs, and we conduct robust statistical tests to confirm the effectiveness of the model.

3.2 Background

This section briefly introduces software defect prediction using ML approaches and the test oracle problem.

3.2.1 Software Defect Prediction

Software Defect Prediction helps to identify defect tendencies in software modules, and there are different approaches to predicting defects. Xu et al. [3] proposed a supervised technique called the Augmented-Code Property Graph-based defect prediction method to predict certain defect types. Wang et al. [4] used supervised gated hierarchical long short-term memory networks which extract semantic features

of abstract syntax trees of source code and traditional features provided by the PROMISE repository. Kumar et al. [56] proposed an unsupervised software defect prediction approach using threshold derivation for clustering and labelling the data points.

Supervised and unsupervised machine learning techniques are the two main types used in software defect prediction [5, 6]. Chen et al.'s [6] study compared supervised and unsupervised machine learning methods and found that unsupervised methods based on the "Lines of Code" metric performed better than or the same as the supervised methods in their study. Li et al. [5] conducted a systematic literature review and found that unsupervised and supervised classifiers performed as well as each other. Xu et al.'s [77] empirical study also indicated that clustering-based unsupervised models are worse than supervised models. Software defect prediction is based on the features extracted from software modules to make recommendations, and "Code metrics" and "Process metrics" are the two commonly used software feature metrics for defect prediction [4, 66]. These metrics collect data from software modules for training the defect prediction models using machine learning [67].

3.2.2 Machine Learning and Test Oracle

Supervised ML technique is learning with labelled data to make predictions. However, labelled software defect data for training can be unavailable [5]. Therefore, unsupervised techniques become popular because it is learning with unlabelled data [5, 6]. Unsupervised ML systems perform as well as the supervised ML systems and clustering-based prediction approaches dominate USDPs [5, 27]. However, USDPs do not have a ground truth to refer to, which is like a test oracle problem to validate the USDPs, and MT is a widely used approach to address the test oracle problem [19-24]. MT has been used to validate the "expected" relationships between inputs and outputs of program executions. These "expected" relationships are expressed as metamorphic relations (MRs), and they are the core element of MT. If the output

results in various software executions violate an MR, then a fault is revealed [19, 21, 24, 31]. In a situation in which software modules have limited historical data and limited experts' or users' inputs, MRs could be challenging to determine [19, 23, 41], and a generic approach can be applied. Zhou et al. [24] has defined Metamorphic Relation Patterns (MRPs) to derive multiple concrete metamorphic relations. Segura et al. [31] presented another list of MRPs to assist in identifying MRs in testing query-based systems. Chua et al. [75] 's study on generic MR generation was based on abstract MRP and symmetry. Studies proved that the μ MT model, which used DMOs to construct MRs, is feasible [41]. Identifying MRs can be achieved by providing a set of DMOs and mapping rules of μ MT [76].

3.3 Motivation

Motivated by the necessity of more advanced testing techniques for validating ML algorithms that suffer from the test oracle problem and situations in which software modules have limited historical data and limited inputs of experts or users, this experiment targets to develop a generic MT model and demonstrate the feasibility of model validating SDPs through a comprehensive scaled experiment. MT is useful in validating ML-based systems; however, the MRs are created mainly through users or experts, which might not be available and might introduce human bias. Thus, we aim to use a generic approach for MR creations to remove relying on users and experts, in order to reduce the heuristic influence on MR creation and demonstrate generic MT's practicality in validating ML-based systems through a comprehensive scaled experiment.

3.4 Proposed Approach

This section describes the validation model and how the generic MT model is designed.

3.4.1 Generic Metamorphic Testing model

This study adapts MT to validate SDPs. In order to establish a generic and adaptable MT model, we followed [19]’s future direction to experiment with the METTLE model on high-dimensional datasets with two non-scalable labels. We identified five types of MRs from the METTLE model [19]. Then, we mapped the first three types of MRs with generic DMOs of μ MT [41, 76] and mapped the rest with our proposed DMOs to establish the generic MRs of the MT model.

Table 3.1: Mapping table of MRs and DMOs.

MR of METTLE	DMO mappings	Source of DMOs
MR1: Manipulating object order	MR1.1 - Swap(x1, x2)	μ MT
MR2: Manipulating the distinctness Among cluster in dataset	MR2.1 - Half(Half(x)) MR2.2 - Half(x) MR2.3 - Double(x) MR2.4 - Double(Double(x))	μ MT
MR3: Manipulating the object density of one or more clusters in the dataset	MR3.1 - x AND Half(Half(x)) MR3.2 - x AND Half(x) MR3.3 - x AND Double(x) MR3.4 - x AND Double(Double(x)) MR3.5 - x AND Half(x) AND Half(Half(x)) MR3.6 - x AND Double(x) AND Double(Double(x)) MR3.7 - x AND Half(x) AND Half(Half(x)) AND Double(x) AND Double(Double(x))	μ MT
MR4: Manipulating attributes	MR4.1 - Remove the highest co-related feature MR4.2 - Remove 10% of total features which are the highest correlated	Proposed
MR5: Manipulating outliers	MR5.1 - $\bar{x} \pm 3\sigma$ MR5.2 - $\bar{x} \pm 4\sigma$ MR5.3 - $\bar{x} \pm 5\sigma$ MR5.4 - $\bar{x} \pm 6\sigma$ MR5.5 - $\bar{x} \pm 3\sigma$ AND $\bar{x} \pm 4\sigma$ MR5.6 - $\bar{x} \pm 3\sigma$ AND $\bar{x} \pm 4\sigma$ AND $\bar{x} \pm 5\sigma$ MR5.7 - $\bar{x} \pm 3\sigma$ AND $\bar{x} \pm 4\sigma$ AND $\bar{x} \pm 5\sigma$ AND $\bar{x} \pm 6\sigma$	Proposed

This adaptation enhances the model’s versatility, allowing users to apply it to numeric data features studied by various machine learning algorithms without understanding the underlying models. The proposed model does not include MR “manipulating the coordinate system” of METTLE because it is challenging to ensure the correctness of transforming the coordinates of high-dimensional datasets and determining the outcomes. In this regard, the proposed model is more adaptable and accessible. This study creates twenty-one individual generic MRs from five types of MRs of METTLE mapping with DMOs of μ MT and our proposed DMOs (Table 3.1).

(1) MR1: Manipulating object order

This relation assumes that the order of objects should not affect the prediction outcomes [19]. This MR could use “Swap” DMO of μ MT [41, 76]. This MR reverses the order of the source dataset to create the follow-up dataset.

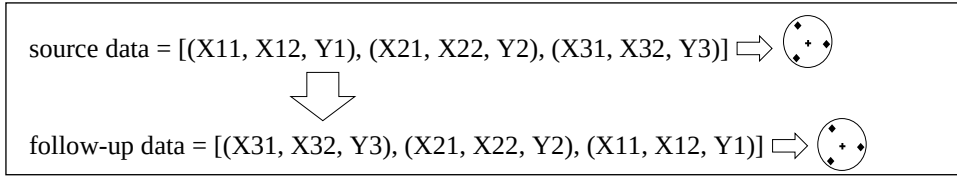


Figure 3.1: Illustration of MR1 data transformation will not affect the clustering results.

(2) MR2: Manipulating the distinctness among cluster in dataset

This relation assumes that shrinking and expanding proportionally to some or all clusters towards to or away from their centroids in the dataset should not affect the prediction outcomes [19].

We chose “Half” and “Double” DMO of μ MT [41, 76], which means half and double all the values of all features, for this MR. In addition, this study applied the same DMO again to increase the levels of shrinking and expanding, which means a quarter times and four times all the values of all features.

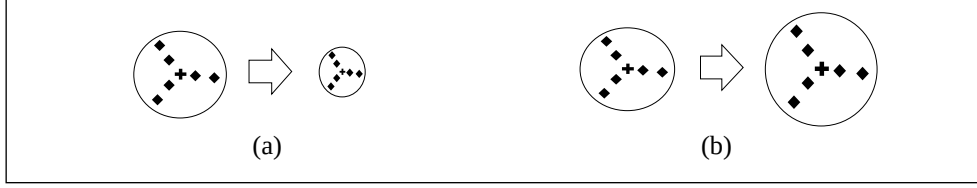


Figure 3.2: (a) Illustration of MR2 shrinking in proportion will not affect the clustering results. (b) Illustration of MR2 expanding in proportion will not affect the clustering results.

(3) MR3: Manipulating the object density of one or more clusters in the dataset

This relation assumes that adding objects into clusters in the dataset to increase the object densities of these clusters should not affect the prediction outcomes [19]. This MR reused the follow-up datasets created in MR2 and added them to the source dataset to create different levels of densities for testing.

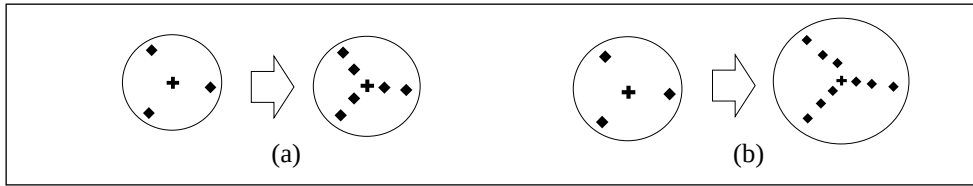


Figure 3.3: (a) Illustration of MR3 increases density with instances that are shrinking in proportion will not affect the clustering results. (b) Illustration of MR3 increases density with instances that expanding in proportion will not affect the clustering results.

(4) MR4: Manipulating attributes matched with removing highly correlated features

This relation assumes a high correlation indicates that an attribute can be considered redundant and, hence, to be removed [19]. This study proposed using the removal of highly correlated features as DMO. Two MRs are proposed. One is to remove the highest correlated attribute in the dataset. The other one is to remove about 10% of total attributes, which are the highest correlated attributes.

(5) MR5: Manipulating outliers

This relation assumes the clustering system will handle outliers by filtering them or assigning new cluster labels; thus, it should not affect the prediction outcomes [19].

Data transformation \ Label	$\bar{x} + 3\sigma$	$\bar{x} - 3\sigma$
+ve label	4	4
-ve label	4	4
Total Outliners	8	8

(a) MR5.1

Data transformation \ Label	$\bar{x} + 3\sigma$	$\bar{x} - 3\sigma$	$\bar{x} + 4\sigma$	$\bar{x} - 4\sigma$
+ve label	2	2	2	2
-ve label	2	2	2	2
Total Outliners	4	4	4	4

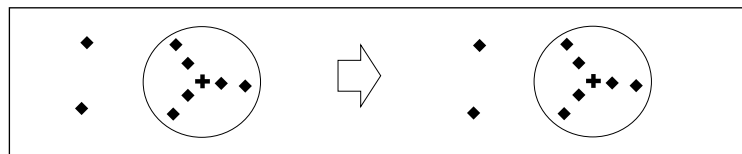
(b) MR5.5

Data transformation \ Label	$\bar{x} + 3\sigma$	$\bar{x} - 3\sigma$	$\bar{x} + 4\sigma$	$\bar{x} - 4\sigma$	$\bar{x} + 5\sigma$	$\bar{x} - 5\sigma$
+ve label	1	1	1	1	1	1
-ve label	1	1	1	1	1	1
Total Outliners	2	2	2	2	2	2

(c) MR5.6

Data transformation \ Label	$\bar{x} + 3\sigma$	$\bar{x} - 3\sigma$	$\bar{x} + 4\sigma$	$\bar{x} - 4\sigma$	$\bar{x} + 5\sigma$	$\bar{x} - 5\sigma$	$\bar{x} + 6\sigma$	$\bar{x} - 6\sigma$
+ve label	1	1	1	1	1	1	1	1
-ve label	1	1	1	1	1	1	1	1
Total Outliners	2	2	2	2	2	2	2	2

(d) MR5.7



(e) outliers will not affect the clustering results

Figure 3.4: Illustration of MR5 (a)MR5.1 (b)MR5.5 (c)MR5.6 (d)MR5.7 (e)Outliers will not affect the clustering results due to the distances

There are two common approaches to classifying outliers: standard deviation with approximately symmetric distributions or interquartile range with skewed distribution [203]. Assuming the data distributions are approximately symmetrical, this study proposed using the mean values and different levels of standard deviation values of features as DMO to create outliers. We created the same number of outliers with positive and negative labels to reduce the effect of imbalanced labels. For example, MR5.1 used transformation $\bar{x} \pm 3\sigma$, which means adding or subtracting three times the standard deviation values of features to the mean values of features, in order to create outliers (Fig. 3.4a). MR5.6 has six data transformations, so we created six lines of outliers with a positive label and six lines of outliers with a negative label (Fig. 3.4c).

3.5 Experiment with Generic Metamorphic Testing model on Validating Machine learning-based Software Defect Detection Systems

This section describes the setup of the experiment. It defines the research objective and research questions of our experiment. Then, it describes the subject systems and data used in the experiment. Thereafter, it discusses detailed experimental design, including environment configuration, measurement, experimental procedure, dataset preparation, and parameter setting. Finally, we present our experimental results and discuss, analyse, and interpret the results for research questions.

3.5.1 Research Questions

The main objective of the experiment is to demonstrate the feasibility of the proposed model for validating USDPs using clustering algorithms and SSDPs. This study aims to answer the following research questions.

RQ1: Could the generic MT model validate SDPs? Different MRs have different impacts on MT validation capacity for different ML algorithms. We aim to evaluate the effectiveness of different MRs in validating different ML algorithms by counting the occurrence of violations related to MRs. These numbers can indicate the effectiveness of the generic MT model.

RQ2: What are the underlying causes of violations? When a violation occurs, we analyse the results with the violation patterns, the characteristics of MRs, and the characteristics of SDPs, in order to seek any relationship that leads to the violations.

RQ3: What are the differences in applying generic MT model to validate USDPs and SSDPs? We further analyse the effectiveness of MT on supervised and unsupervised ML algorithms by comparing the differences in prediction inconsistencies and accuracies between USDPs and SSDPs.

3.5.2 Subject Machine learning algorithm

[78, 79] successfully applied MT to validate unsupervised ML algorithms; however, their studies only involved three ML algorithms. This study explores the feasibility of applying the proposed model to validate different SDPs utilizing different ML algorithms. Thus, our experiment involved eighteen USDPs and two SSDPs utilizing fourteen different ML algorithms (Table 3.2). They were obtained from the “scikit-learn” library, a popular ML library for Python, as [80] mentioned in their study. Version 0.24.2 of the “scikit-learn” library is used in this experiment.

3.5.2.1 Unsupervised Software Defect Prediction models (USDPs)

USDPs are using some of the most widely used unsupervised clustering algorithms [5, 56, 77, 81, 82], and we occupied clustering classifications of Xu et al. [77] to group the subject USDPs as the following.

Model-Based Clustering (MBC) assumes the probability model for each cluster and arranges instances to the cluster that best fits the model [77]. This study selected two expectation-maximization (EM) methods, including the “Gaussian Mixture Model” (GMM) and the “Bayesian Gaussian Mixture Model” (BayGM).

GMM uses a probabilistic assignment of data points to clusters where each cluster has its separate Gaussian distribution [82]. In the “scikit_learn” library, the “GaussianMixture” function has a mandatory option for choosing the type of covariance [83, 84]. This study treats each GMM model with different covariance options as separate SDP, so there are three GMM models. They are GMM with “diagonal” covariance matrix option (GMM(diag)), GMM with “spherical” covariance matrix option (GMM(spherical)), and GMM with “full” covariance matrix option (GMM(full)).

BayGM is a variant of GMM and uses the Bayesian inference framework to estimate the posterior distribution of the parameters and the hyperparameters of the distribution [85]. In the “scikit_learn” library, the “BayesianGaussianMixture” function has a mandatory option for choosing the type of covariance [83, 86]. This study treats each BayGM model with different covariance options as separate SDP, so there are three BayGM models. They are BayGM with “diagonal” covariance matrix option (BayGM(diag)), BayGM with “spherical” covariance matrix option (BayGM(spherical)), and BayGM with “full” covariance matrix option (BayGM(full)).

Partition-Based Clustering (PBC) randomly selects partitions of instances as clusters and assigns the remaining instances to clusters whose centre point is closest to those instances. It updates the centre point of each cluster and relocates the clusters of other instances iteratively until the centre points of the clusters remain unchanged [77]. This study selects “K-Means”, “Mini-Batch K-Means”, and “Bisecting K-Means” clustering methods.

K-Means uses squared Euclidean distance as the similarity measure to select instances in the clusters. The number of clusters is typically chosen by the user, and it aims to minimize the sum of the squared distances between each data point and its assigned cluster centre [77, 81]. The “scikit_learn” library has a mandatory EM algorithm option for the K-Means model [87]. This study treats each K-Means model with different EM algorithm options as separated SDP, so there are two K-Means models. They are the K-Means model using the “lloyd” EM algorithm (Kmean (full)) and the K-Means model using the “elkan” EM algorithm (Kmean(elkan)) [87, 88].

Mini-Batch K-Means (MiniKM) is a variant of K-Means that uses shorter batches of datasets to calculate similarity, in order to reduce the computation time [77].

Bisecting K-Means is a variant of K-Means and works by recursively dividing clusters into two sub-clusters until the desired number of clusters is reached [89]. The “scikit_learn” library has a mandatory EM algorithm option for the Bisecting K-Means model [90]. This study treats each Bisecting K-Means model with different EM algorithm options as separated SDP, so there are two Bisecting K-Means models, and they are Bisecting K-Means model using “lloyd” EM algorithm (BisectKM(full)), and Bisecting K-Means model using “elkan” EM algorithm (BisectKM(elkan)) [88, 90].

Graph-Theory-Based Clustering (GTBC) constructs a weighted graph where each node represents an instance, and the weight of the edge denotes the similarity measure of its two nodes. It iteratively divides graphs until it separates all instances into non-overlapped clusters [77]. This study selected “Spectral Clustering” (Spectral) and “Affinity Propagation Clustering” (Affinity).

Spectral transforms the data into a lower-dimensional space using the eigenvectors of a similarity matrix. By using the similarity matrix, it constructs a connected graph, in which every pair of points is connected by a weighted edge, and treats the clustering as a graph partitioning problem [77, 91, 92].

Affinity regards all the data points as potential cluster centres and the negative value of the Euclidean distance between two data points as the affinity. The core idea means that the larger the sum of the affinity of one data point, the higher probability of this data point being the cluster centre [5, 77, 91, 93].

Hierarchy-Based Clustering (HBC) creates a hierarchical decomposition of the data. This study selected “Agglomerative Clustering” (Agglo) and “Balanced iterative reducing and clustering using hierarchies” (Birch) clustering [77].

Agglo uses bottom-up decomposition to build clusters. It firstly treats each instance as a separate cluster and iteratively merges each instance to the closest cluster. Secondly, the process continues until all instances are merged into clusters or a predefined condition is met [77].

Birch, on the other hand, uses top-down decomposition to find clusters. It, firstly, treats all instances as one initial cluster and iteratively splits the cluster into smaller ones. Secondly, this process continues until each instance belongs to one cluster or a predefined condition is met [77].

Density-Based Clustering (DBC) uses the notion of data distribution density to form dense regions as clusters [77]. This study selected “Density-Based Spatial Clustering of Applications with Noise” (DBscan), “Ordering Points to Identify the Clustering Structure” (Optics), and “MeanShift” clustering (Meanshift) [91].

DBscan clusters each instance based on their proximity and density. Although the performance of DBscan is related to the input parameters, DBscan starts by randomly selecting one data point and expands its neighbourhood until it contains the minimum number of data to form a cluster [77, 94, 95].

Optics builds a reachability graph that represents the density-based connectivity of the data points in the feature space. Then it creates a hierarchical clustering of the data points based on their reachability distances. This way allows Optics to identify

clusters of different densities and shapes, while being robust to noise and outliers [91, 94, 95].

Meanshift assumes that datasets of different cluster classes conform to different probability density distributions. It finds clusters by iteratively moving the centroid of a kernel density estimate towards high-density regions by calculating the mean of offset of the current data point to find the direction where the density of any sample point increases the most [77, 91, 92].

3.5.2.2 Supervised Software Defect Prediction models (SSDPs)

We include two SSDPs based on two widely used and proven machine learning algorithms [6, 67, 96] in this study for the purpose of comparing the results with USDPs. They are as follows:

Random Forests (RF). There are many supervised clustering-based algorithms for software defect predictions, and RF is one of the most used and well-performed algorithms in this area [6, 67, 96, 97]. RF uses hyper bagging or averaging methods of the decision trees to give prediction results to improve performance and reduce the chance of over-fitting problems [6, 98].

K-Nearest Neighbours (KNN) is another type of SSDP. KNN is a commonly used supervised clustering-based algorithm for software defect predictions [67, 96]. It uses the distance function to measure the similarity between instances and assigns the most common class of processing instance's "k" nearest neighbours to the processing instance [98-100]. "k" is the most crucial hyper-parameter in KNN as this could lead to the model to under-fitting or over-fitting the data [98].

Table 3.2: List of subject machine learning algorithms

Type	Machine Learning family	Machine learning algorithm	Abbreviation
Unsupervised	Model-Based	Gaussian Mixture Model with "diagonal" covariance matrix option	GMM(diag)

	Clustering (MBC)	Gaussian Mixture Model with “spherical” covariance matrix option	GMM(spherical)
		Gaussian Mixture Model with “full” covariance matrix option	GMM(full)
		Bayesian Gaussian Mixture Model with “diagonal” covariance matrix option	BayGM(diag)
		Bayesian Gaussian Mixture Model with “spherical” covariance matrix option	BayGM(spherical)
		Bayesian Gaussian Mixture Model with “full” covariance matrix option	BayGM(full)
	Partition-Based Clustering (PBC)	K-Means model using “lloyd” EM algorithm	Kmean(full)
		K-Means model using “elkan” EM algorithm	Kmean(elkan)
		Mini-Batch K-Means	MiniKM
		Bisecting K-Means model using “lloyd” EM algorithm	BisectKM(full)
		Bisecting K-Means model using “elkan” EM algorithm	BisectKM(elkan)
	Graph-Theory-Based Clustering (GTBC)	Spectral Clustering	Spectral
		Affinity Propagation Clustering	Affinity
	Hierarchy-Based Clustering (HBC)	Agglomerative Clustering	Agglo
		Balanced iterative reducing and clustering using hierarchies	Birch
	Density-Based Clustering (DBC)	Density-Based Spatial Clustering of Applications with Noise	DBscan
		Ordering Points to Identify the Clustering Structure	Optics
		MeanShift clustering	Meanshift
Supervised		Random Forests	RF
		K-Nearest Neighbours	KNN

3.5.3 Subject Dataset

This study has chosen publicly available and widely used software defect datasets of NASA¹ and PROMISE² data repositories as subject datasets [96, 101]. NASA

¹ NASA Datasets were downloaded on 2022/07/11 and from <https://github.com/klainfo/NASADefectDataset>

² PROMISE datasets were downloaded on 2022/07/11 and from https://figshare.com/articles/dataset/Software_Defect_Prediction_Dataset/13536506

datasets were created based on software metrics and process metrics, which are used to estimate software stages of enhancement, including Cohesion, Coupling and Complexity [66]. PROMISE data repository is designed for “research to predict effort and defect of a software and then developed to be model-based requirements engineering and value-based software engineering” [101]. Thus, they serve the intent of this study, and we have selected five datasets from NASA and fifteen datasets from PROMISE because they are all numeric features (which suits our model and the mapped data mutation operators) and contain less than 1200 instances (which fits the limited computer resources we have) (Table 3.3).

Table 3.3: Information of subject datasets.

Name of dataset	Repository	Number of features	Total number of instances
CM1	NASA	37	330
KC3	NASA	39	195
MW1	NASA	37	250
PC1	NASA	37	680
KC1	NASA	21	1165
ant-1.7	PROMISE	21	745
camel-1.6	PROMISE	21	965
data_arc	PROMISE	21	225
data_prop-6	PROMISE	21	645
data_ivy-2.0	PROMISE	21	355
data_redaktor	PROMISE	21	175
jedit-4.2	PROMISE	21	370
log4j-1.1	PROMISE	21	110
lucene-2.0	PROMISE	21	195
poi-2.0	PROMISE	21	315
synapse-1.2	PROMISE	21	260
velocity-1.6	PROMISE	21	230
xalan-2.4	PROMISE	21	725
xerces-1.2	PROMISE	21	440
xerces-1.3	PROMISE	21	455

3.5.4 Environment

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “scikit_learn”, for machine learning algorithms. Standard parameters were applied to all SDPs, with the cluster number set to two, and the rest options were set to default values (Appendix A). As the datasets are high-dimensional, in order to improve the speed without compensating the data integrity, this experiment used “Principal component Analysis” (PCA) for dimensionality reduction and for preserving as much variability as possible, in order to give the best possible representation of the high-dimensional datasets [102]. We used the PCA function from the “scikit_learn” library to reduce datasets over thirty features into ten principal components to preserve over 85% of the variance in the data explained by ten principal components. Additionally, we used the PCA function to reduce datasets with twenty-one features into five principal components to preserve over 85% of the variance in the data explained by five principal components.

3.5.5 Measurement

This experiment uses the “Re-clustering percentage” (RP) in the METTLE study to “measure the inconsistency between a source output and its corresponding follow-up output” [19]. Source outputs were the prediction outcomes of SDPs trained by the source datasets, and follow-up outputs were those of SDPs trained by follow-up datasets (section 3.5.6). If there is no violation between source output and follow-up output, RP is zero, and the SDP is insensitive to the associated MR, which means the MR could be used for validation.

Let source dataset be $D = [x_1, x_2 \dots x_n]$ and follow-up dataset be $D_{mr} = [x_{m1}, x_{m2} \dots x_{mn}]$. SDP was trained by D to be SDP_d . SDP was trained by D_{mr} as SDP_{mr} . Let test dataset be $D_{test} = [x_{t1}, x_{t2} \dots x_{tz}]$. The prediction outcomes of SDP_d with D_{test}

be $Y_d = [y_{dt1}, y_{dt2} \dots y_{dtz}]$ and the prediction outcomes of SDP_{mr} with D_{test} be $Y_{mr} = [y_{mrt1}, y_{mrt2} \dots y_{mrtz}]$. Let *diff* be the number of cases with cluster label differences between Y_{mr} and Y_d . RP is calculated as Equation (3.1).

$$RP = \text{diff} / D_{test} \quad (3.1)$$

The above calculation indicates that if *diff* is zero, which means there is no difference between SDP_d and SDP_{mr} , RP will be zero, which is considered as zero violation to this MR. If *diff* is larger than zero, then RP will be larger than zero, and violations have occurred. Regarding validating clustering systems, if RP is zero, no violation is caused by MR and vice versa.

This study also employs the “Violation rate” (VR) in the METTLE study to measure how many trials have violation occurred [19]. For example, if the RP of the trial is not zero, the trial is counted as a violated trial. This measurement is used with RP to indicate how sensitive the models react to MR. If both VR and RP are high, this could explain why the model is highly sensitive to the MR. On the other hand, if VR is high and RP is low, this could be interpreted as the model is moderately sensitive to the MR. Let the total number of test trials be T_{total} and the violated test trials be $T_{violated}$. VR is calculated as Equation (3.2).

$$VR = T_{violated} / T_{total} \quad (3.2)$$

As the subject datasets are labelled, we can employ the prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) [73]. These measurements can show any relationship between MRs and prediction accuracy. Let T_p be true positives that are the actual positives and correctly predicted, T_n be true negatives that are the actual negatives and correctly predicted, F_p be false positives that are the actual negatives and wrongly predicted as positives, and F_n be false negatives are the actual positives and wrongly predicted negatives. ACC is calculated as Equation (3.3), and MCC is calculated as Equation (3.4)

$$ACC = (T_p + T_n) / (T_p + T_n + F_p + F_n) \quad (3.3)$$

$$MCC = \frac{(T_p \times T_n - F_p \times F_n)}{\sqrt{((T_p + F_p) \times (T_p + F_n)) \times ((T_n + F_p) \times (T_n + F_n))}} \quad (3.4)$$

Apart from the above measurements, this study also examined the violation pattern related to MRs to understand better the underlying cause of the violation and the behaviour of SDPs. Inspired by the study of Xie et al. [19], this study has defined categories of violation patterns (Table 3.4).

Table 3.4: Violation Pattern of Relabelling after apply MRs

Violation Pattern	Description
Reverse	Over 75% of results are relabelled
New Cluster created	New Cluster(s) is created
Noise	Only outliers are relabeled
No change	No relabelling
Border	Relabeling occurred on the edge between clusters
Merge and Split	Clusters are merged or cluster(s) is split
Less Cluster	Less cluster(s) is created

3.5.6 Implementation

In the context of scenarios characterized by the scarcity of labelled datasets, USDPs are applied. Consequently, this study assumes that the testing datasets do not have labels, and it is difficult to validate with unlabelled testing datasets solely. Thus, we trained the SDPs with data generated by MRs, which should not affect the prediction outcomes (section 3.4). We then used the outcomes from various trained SDPs to assess the consistency and validate the SDPs. Our experiment includes four steps as follows:

Step 1. We use twenty subject datasets (section 3.5.3) as source datasets and randomly select twenty per cent of the data of each source dataset to create test datasets for comparison used in step 4. Test datasets will not be used for training.

Step 2. We apply the proposed MRs to source datasets (section 3.4) and generated twenty-one follow-up datasets for each subject dataset.

Step 3. SDPs train with source datasets and become “Source SDPs”. SDPs train with follow-up datasets and become “Follow-up SDPs”.

Step 4. “Source SDPs” make predictions on test datasets to create “source outputs”. “Follow-up SDPs” make predictions on the same test datasets to create “follow-up outputs”. Then, we compare source and follow-up outputs and analyse their differences. A failure is revealed if the relation is violated, which means differences occur between the outputs (Fig. 3.5).

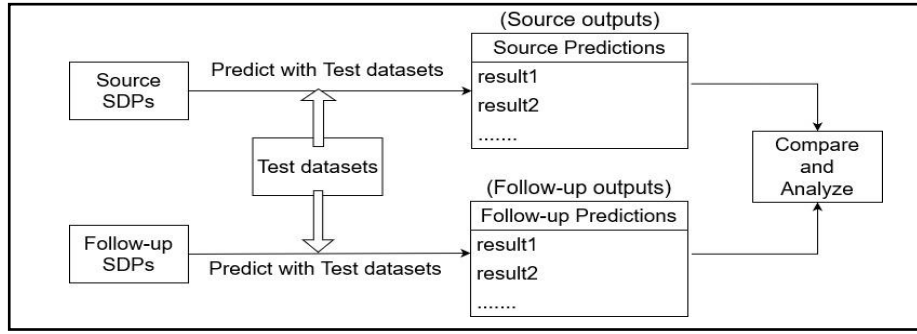


Figure 3.5: Illustration of the experiment

Twenty subject SDPs were trained with twenty subject source datasets, so there were 400 source SDPs. Twenty-one MRs were applied to twenty subject datasets, so there were 420 follow-up datasets. With twenty subject SDPs, there are 8400 follow-up SDPs. We conducted five trials on the test datasets with 400 source SDPs resulting in 2000 trials for source outputs. We also conducted five trials on the test datasets with 8400 follow-up SDPs, making 42000 trials for follow-up outputs. With respect to twenty subject SDPs, we conducted 44000 trials in this study.

3.5.7. Result and Discussion

We first studied the violations of all MRs with all subject SDPs to evaluate the effectiveness of our proposed model (RQ1). Then, we further analysed the violation patterns to seek more understanding of the causes (RQ2). At last, we analysed the

performances between the SSDPs and USDPs to examine the relationship between labelled datasets and our proposed model (RQ3).

3.5.7.1 RQ1: Could generic MT model validate SDPs?

To answer this RQ, we reviewed the results of “Zero violation with all MRs”, “Zero violation with MR2”, “Violated all MRs”, and “Violated MR1, MR3, MR4 and MR5”.

3.5.7.1.1 Zero violation with all MRs

Table 3.6 shows the average RP, VR, ACC and MCC values of all SDPs in this study. Agglo and DBscan were two USDPs that had zero RP value (zero violation) with all twenty-one MRs and all twenty test datasets. These two USDPs are highly adaptable and robust to our model. However, the average ACC of DBscan was only 22.33%, and the average ACC of Agglo was 63.85%, with the third-highest MCC of all USDPs (Table 3.6). Agglo was the only SDP that made predictions better than random guesses (over 50%) and was robust to our model.

Agglo and DBscan use the “bottom-up” approach and do not randomly select an initial sample of points as they treat all single instances as clusters in the beginning [77, 111]. As long as the transformations do not alter the values of part of features and do not alter density distributions, Agglo and DBscan can produce consistent results.

MR1 reverses the order of data without changing the values of features or density distributions. MR2 only shrinks and expands values of all the features proportionally, so there is no change in values of part of features or density distributions of data. MR3 adds the datasets created by MR2 to source datasets to create different levels of densities. Since MR2 does not change the values of some features or density distributions, adding them to the source data should not affect their performance. MR4 removes highly correlated features, which indicates that an attribute can be

considered redundant from datasets. As they are considered redundant, removing them should not affect the values of overall features or density distributions of data. MR5 adds different combinations of outliers to data, and this study created outliers of at least three standard deviations of all values of all features. Agglo and DBscan can detect added outliers, making both algorithms insensitive to MR5. Therefore, the proposed model can serve as a validation tool for SDPs using Agglo and DBscan with high-dimensional data.

3.5.7.1.2. Zero violation with MR2

Table 3.7 shows that twelve out of twenty SDPs did not violate MR2. Fig. 3.6a shows MR2 had a much lower average RP than the other MRs, as the maximum average RP of MR2 were below all the lower quartiles of MR1, MR3, MR4 and MR5. To examine whether the differences were significant, we ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in average RP between Agglo (zero-violated SDP) and other SDPs, which violated MR2. Table 3.5 shows that only the mean of differences in average RP of Affinity was significantly different from Agglo. Thus, twelve SDPs had zero violation with MR2, and seven SDPs did not significantly violate MR2; this concludes that MR2 is a must property of the proposed model.

Table 3.5: Multiple paired t-tests with significance level at 0.01 for mean of difference in average RP values of MR2 between Agglo and Affinity which were significantly different.

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs Affinity	MR2.1	0.0099	YES
	MR2.2	0.0062	YES
	MR2.3	0.0268	NO
	MR2.4	0.0028	YES

Table 3.6: Average RP, VR, ACC and MCC values of SDPs with all MRs and all test datasets.

	GMM (diag)	GMM (sph.)	GMM (full)	Bay-GM (diag)	Bay-GM (sph.)	Bay-GM (full)	Kmean (full)	Kmean (elkan)	Mini-KM	Bisect-KM (full)	Bisect-KM (elkan)
RP	39.2%	42.6%	39.6%	23.4%	21.4%	37.5%	34.1%	34.1%	32.6%	36.1%	36.3%
VR	80.5%	75.0%	75.2%	78.3%	74.8%	79.3%	70.9%	71.0%	71.0%	71.4%	71.2%
ACC	52.6%	53.7%	52.1%	61.9%	65.5%	52.9%	61.0%	60.9%	60.4%	60.0%	59.6%
MCC	3.7%	4.7%	5.3%	13.5%	15.6%	10.6%	8.7%	8.6%	6.5%	7.7%	7.7%
	Mean-shift	Spectral	Affinity	Agglo	Birch	Optics	DBscan	KNN	RF		
RP	6.1%	2.4%	53.4%	0.0%	36.8%	7.9%	0.0%	8.8%	10.5%		
VR	79.5%	5.0%	53.5%	0.0%	67.0%	30.0%	0.0%	78.0%	71.5%		
ACC	75.4%	65.4%	36.7%	63.8%	60.6%	21.5%	22.3%	84.7%	82.4%		
MCC	14.8%	10.4%	0.0%	11.4%	0.5%	3.0%	5.7%	35.6%	29.2%		

Table 3.7: Average RP values of MR2 of SDPs with all test datasets.

MR	GMM (diag)	GMM (sph.)	GMM (full)	Bay-GM (diag)	Bay-GM (sph.)	Bay-GM (full)	Kmean (full)	Kmean (elkan)	Mini-KM	Bisect-KM (full)	Bisect-KM (elkan)
MR2.1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.00%	1.00%	0.00%	4.00%	1.97%
MR2.2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	2.00%	2.00%	0.00%	3.00%	3.98%
MR2.3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.00%	0.00%	2.00%	1.03%
MR2.4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.00%	2.00%	0.00%	4.00%	2.07%
MR	Mean-shift	Spectral	Affinity	Agglo	Birch	Optics	DBscan	KNN	RF		
MR2.1	0.00%	2.68%	28.00%	0.00%	0.00%	9.25%	0.00%	0.00%	0.21%		
MR2.2	0.00%	2.72%	32.00%	0.00%	0.00%	9.25%	0.00%	0.00%	0.09%		
MR2.3	0.00%	3.16%	22.00%	0.00%	0.00%	9.25%	0.00%	0.00%	0.09%		
MR2.4	0.00%	2.32%	37.00%	0.00%	0.00%	9.25%	0.00%	0.00%	0.16%		

3.5.7.1.3. Violated all MRs

Three SDPs violated all MRs and they are Spectral, Optics and Affinity. Spectral violated only the test dataset of MW1, and the average RP in each MR is from 36% to 63%. We ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and Spectral, and test results show the differences were not significant with all MRs. Thus, we conclude that these violations are not related to MRs.

Table 3.8: Average RP of Optics with violated test datasets.

Test dataset	MR1	MR2.1	MR2.2	MR2.3	MR2.4	MR3.1	MR3.2	MR3.3	MR3.4	MR3.5	MR3.6	MR3.7
CM1	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%
PC1	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%
KC1	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%
ant-1.7	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%
camel-1.6	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%
data_prop-6	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%
xalan-2.4	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%
Test dataset	MR4.1	MR4.2	MR5.1	MR5.2	MR5.3	MR5.4	MR5.5	MR5.6	MR5.7			
CM1	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%			
PC1	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%	16.9%			
KC1	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%	27.0%			
ant-1.7	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%	24.8%			
camel-1.6	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%	45.6%			
data_prop-6	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%	25.6%			
xalan-2.4	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%	42.1%			

Optics violated all MRs with seven test datasets; each test dataset was at the same RP as all MRs (Table 3.8). So, we conclude that the seven datasets cause the violations. Affinity was sensitive to all MRs. Fig. 3.6b shows Affinity violated all MRs at an extensive range of average RP. We further studied the violation patterns of Affinity. We found that the predictions of all trials are either classifying all instances as class zero or class one, which leads to either 100% or 0% RP. That explains why most of the interquartile ranges of MRs in Fig. 3.6b are 100%.

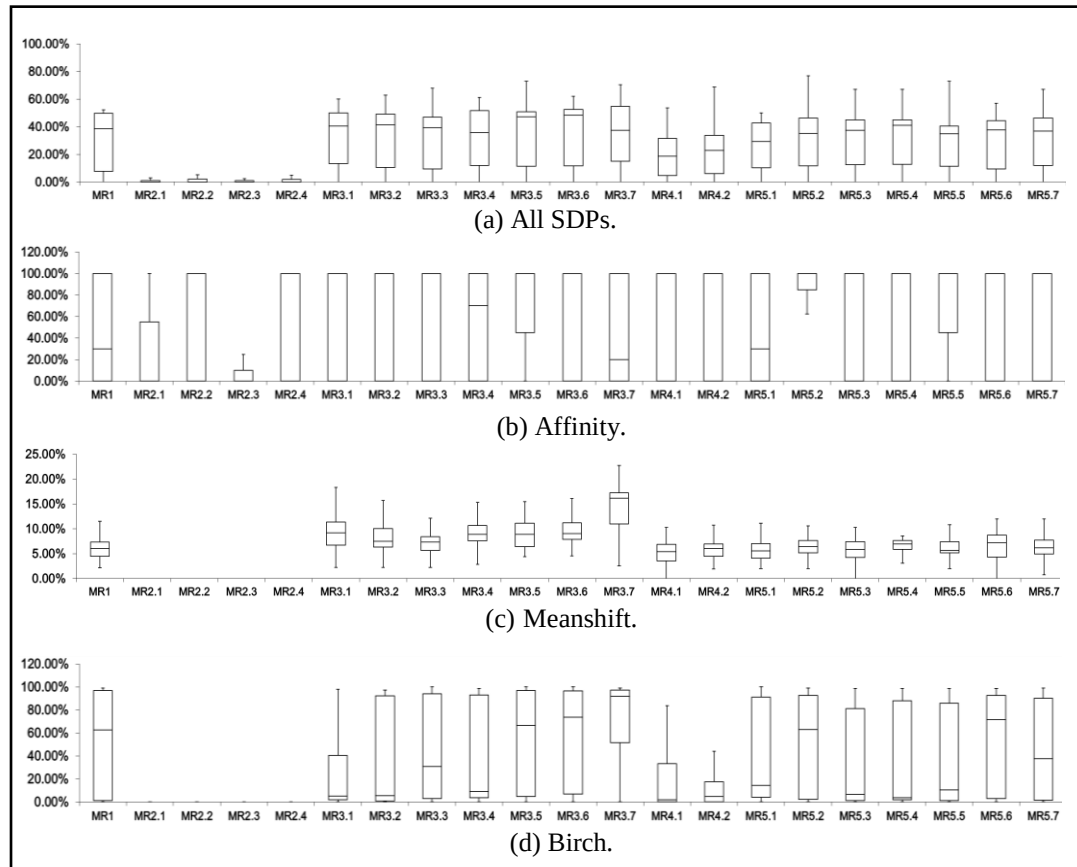
3.5.7.1.4. Violated MR1, MR3, MR4 and MR5

Fifteen SDPs violated all MRs except MR2, including MBC algorithms, PBC algorithms, Meanshift, Birch, K-Nearest Neighbour, and Random forest.

Meanshift had moderate average RP (Fig. 3.6c). We studied the violation patterns of Meanshift and found that most violation patterns had new clusters created for predictions after data transformation. Thus, except MR2, other MRs caused Meanshift to consider instances in the test dataset with new clusters.

Birch was sensitive to all MRs except MR2 (Fig. 3.6d). We studied the result of Birch and found most RP of all trials are either below twenty per cent or above eighty per cent, so this explains why most of the interquartile ranges are large (Fig. 3.6d). We further studied the violation patterns of Birch and found that most violation patterns are “Reverse”, which means, except MR2, data transformations caused Birch to change predictions.

The average RP of K-Nearest Neighbour and Random Forrest are moderate (Fig. 3.6e and Fig. 3.6f). We discovered their violation patterns were mostly “Border” and “Merge and Split” cases, which means the data transformations changed their predictions on instances that are on the decision edge of class zero and class one. We ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and RF and between Agglo and KNN. The results showed that all the differences are significant with MR1, MR3, MR4 and MR5.



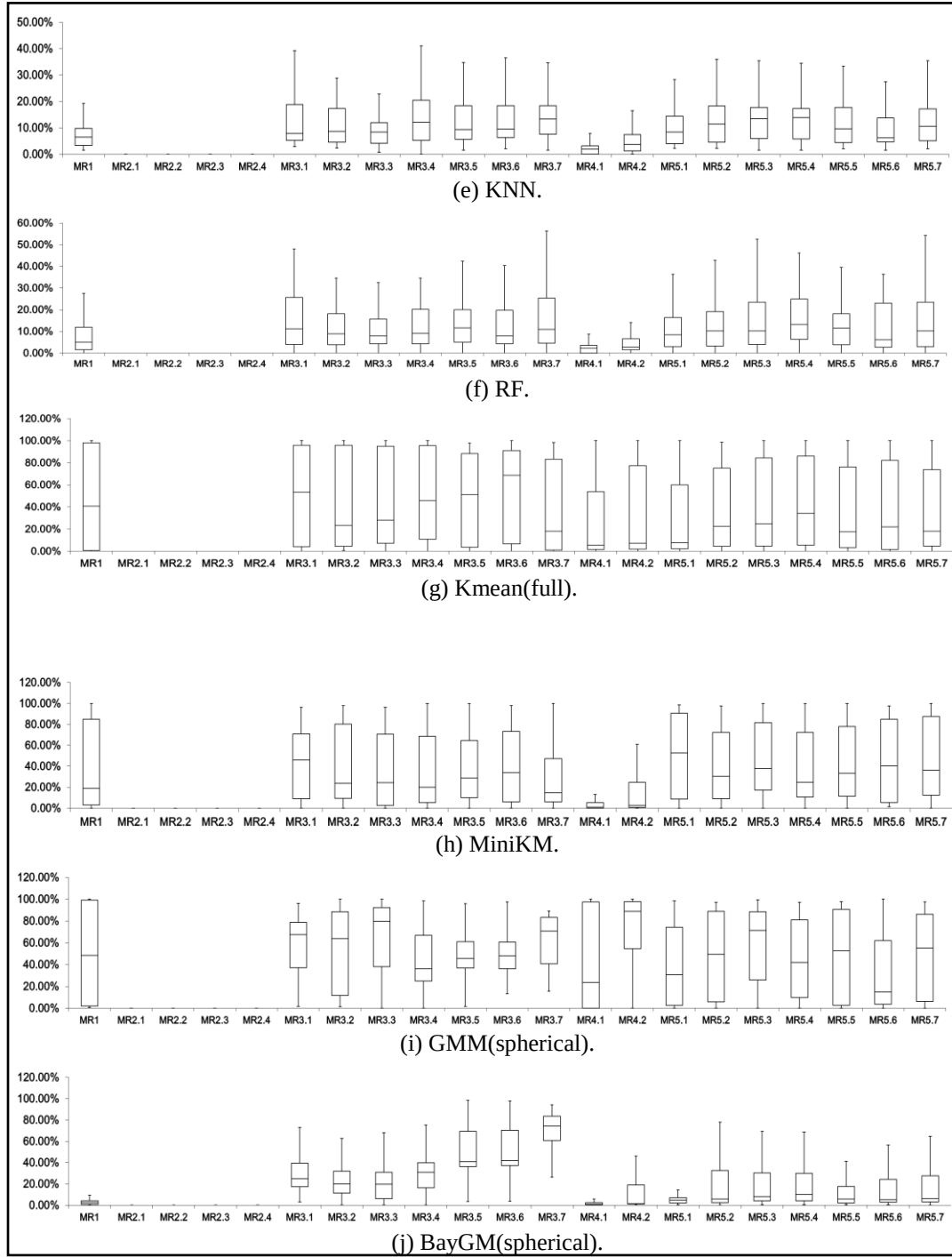


Figure 3.6: Box and whisker diagram of average RP values of all datasets with (a)all SDPs (b)Affinity (c)Meanshift (d)Birch (e)KNN (f)RF (g)Kmean(full) (h)MiniKM (i)GMM(spherical) (j)BayGM(spherical).

Kmean(full) (Fig. 3.6g), Kmean(elkan), BisectKM(full) and BisectKM(elkan) were sensitive to MR1, MR3, MR4 and MR5. We discovered that Kmean(full), Kmean(elkan), BisectKM(full) and BisectKM(elkan) had mostly “Reverse” violation patterns with MR1, MR3 and MR5, and had “Border” and “Reverse” violation patterns with MR4. We also ran multiple paired t-tests with a significance level of 0.01 for the mean of differences in RP between Agglo and PBC algorithms, and all the differences are significant, which matched the violation patterns, except MiniKM in MR4 (Table 3.9).

MiniKM had a lower average RP with MR4 than with other MRs (Fig. 3.6h). It had mostly “Reverse” violation patterns with MR1, MR3 and MR5 and “No change” patterns with MR4.1 and MR4.2. These results align with MiniKM, which did not significantly violate MR4 (Table 3.9). Thus, PBC algorithms were sensitive to MR1, MR3 and MR5, and MiniKM did not significantly violate MR2 and MR4.

Table 3.9: Multiple paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and MiniKM.

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs MiniKM	MR1	0.000404	Yes
	MR3.1	0.000018	Yes
	MR3.2	0.000072	Yes
	MR3.3	0.000416	Yes
	MR3.4	0.000480	Yes
	MR3.5	0.000080	Yes
	MR3.6	0.000098	Yes
	MR3.7	0.000608	Yes
	MR4.1	0.064163	NO
	MR4.2	0.010795	NO
	MR5.1	0.000048	Yes
	MR5.2	0.000063	Yes
	MR5.3	0.000014	Yes
	MR5.4	0.000042	Yes
	MR5.5	0.000058	Yes
	MR5.6	0.000036	Yes
	MR5.7	0.000024	Yes

Table 3.10: Multiple paired t-tests with significance level at 0.01 for mean of difference in RP values of MR1, MR3, MR4 and MR5 between Agglo and BayGM(spherical).

Paired T-test group	MR	P(T<=t) 2 tail (alpha=0.01)	Reject H0: mean diff. = 0 (p-value < 0.01)
Agglo vs BayGM (spherical)	MR1	1.35E-01	NO
	MR3.1	8.95E-06	Yes
	MR3.2	1.58E-04	Yes
	MR3.3	5.03E-04	Yes
	MR3.4	9.63E-06	Yes
	MR3.5	2.76E-08	Yes
	MR3.6	2.25E-08	Yes
	MR3.7	2.58E-11	Yes
	MR4.1	7.09E-02	NO
	MR4.2	2.50E-02	NO
	MR5.1	8.60E-03	Yes
	MR5.2	3.42E-03	Yes
	MR5.3	1.95E-03	Yes
	MR5.4	1.32E-03	Yes
	MR5.5	4.07E-03	Yes
	MR5.6	3.61E-03	Yes
	MR5.7	1.52E-03	Yes

GMM(diag), GMM(spherical) (Fig. 3.6i), GMM(full), BayGM(diag) and BayGM(full) were sensitive to MR1, MR3, MR4 and MR5. We discovered that GMM(diag), GMM(spherical), and GMM(full) had mostly “Merge and split” and “Reverse” violation patterns with MR1, MR3, MR4 and MR5. BayGM(diag) mostly had “No Change” and “Border” patterns with MR4, and “Merge and split” patterns with MR1, MR3, and MR5. BayGM(full) mostly had “Merge and split” patterns with MR1, MR3, MR4, and MR5. We ran multiple paired t-test results for the mean of differences in RP between Agglo and MBC algorithms. All the differences are significant, even with BayGM(diag), except BayGM(spherical) with MR1 and MR4 (Table 3.10).

BayGM(spherical) had lower average RP with MR1, MR4.1 and MR5.1 than with other MRs (Fig. 3.6j). BayGM(spherical) has mostly “Merge and split” patterns with MR3 and MR5, mostly “No Change” and “Border” patterns with MR1 and MR4. The patterns match the multiple paired t-test results (Table 3.10), which showed that BayGM(spherical) did not significantly violate MR1 and MR4. Thus, MBC

algorithms are sensitive to MR1, MR3, and MR5, except MR2 and BayGM(spherical), which did not significantly violate MR1, MR2, and MR4.

Answering RQ1: Agglo and DBscan did not violate any MR of the generic MT model. Nineteen SDPs did not significantly violate MR2, four SDPs did not significantly violate MR4, and three SDPs did not significantly violate MR1. Thus, results conclude that the generic MT model can be used as a validation tool for SDPs, and MR2 is a must property of the generic MT model. In addition, the average ACC of DBscan was below 50% (Table 3.6), so Agglo is the only SDP insensitive to all MRs and can make predictions better than random guesses (over 50%) in the study.

3.5.7.2 RQ2: What are the underlying causes of violations?

This section discusses the three underlying causes of violations with the violated SDPs.

3.5.7.2.1 Caused by characteristics of data

Results showed that the characteristics of data cause violations for Spectral and Optics. Spectral had violations with all MRs in the test dataset generated from dataset MW1 only. We compared dataset MW1 with another two zero-violated datasets, which were also from NASA and had the same features and similar size, PC1 and CM1 (Table 3.3).

We ran multiple paired t-tests with a significance level 0.01 between those three source datasets. We found that the values of the twelve features are significantly different between PC1 and MW1, between CM1 and MW1, and not significantly different between PC1 and CM1. In other words, the values of these twelve features in MW1 significantly differ from those in PC1 and CM1. Table 3.11 shows the statistic of the most different feature between “CM1 and MW1”, and “PC1 and MW1”. The ranges of values in PC1 and CM1 are more extensive than in MW1,

which means the variations of values in PC1 and CM1 are more significant than those in MW1 (Table 3.11).

Table 3.11: The statistic of the most different feature “HALSTEAD_DIFFICULTY” between CM1, PC1 and MW1.

	HALSTEAD_ DIFFICULTY (PC1)	HALSTEAD_ DIFFICULTY (CM1)	HALSTEAD_ DIFFICULTY (MW1)
Mean	20.3746	20.0586	12.3334
Standard Error	0.6747	0.8351	0.5187
Median	15.8400	15.2000	10.2100
Mode	16.0000	12.0000	5.5000
Standard Deviation	17.5820	15.1011	8.2006
Sample Variance	309.1259	228.0419	67.2501
Range	268.6600	124.2300	45.0300
Minimum	2.0000	2.3300	2.5000
Maximum	270.6600	126.5600	47.5300
Count	679	327	250

As the variations of values in the twelve features of MW1 were smaller than PC1 and CM1, the dataset of MW1 was more difficult for Spectral to form an accurate “similarity matrix of the data to construct a connected graph and treating clustering as a graph partitioning problem” [77] than PC1 and CM1. If the ranges and standard deviations of feature values in the datasets are too small, the distances between data points will be very short. This could lead Spectral to consider those data points as similar and belonging to the same cluster, when in fact they are different. Thus, we conclude that Spectral violated all MRs with the test dataset of MW1 because the variations of values of features in dataset MW1 are too small for Spectral to form an accurate similarity matrix for classification.

Optics had violations with seven test datasets and their average RP were the same with all MRs in each of the seven test datasets (Table 3.8). Violated test datasets are generated from datasets CM1, PC1, KC1, ant-1.7, camel-1.6, data_prop-6 and xalan-2.4 and six of them were the largest datasets in the study (Table 3.3). CM1 was the smallest of the seven violated test datasets, with only 330 instances and 37 features

(Table 3.3). Table 3.8 shows that the RP of CM1 was only 3.03% across all MRs, which is also the smallest, and the rest are from 24.83% to 45.6%.

PROMISE dataset xerces-1.3 has fewer instances than PROMISE dataset data_prop-6 (Table 3.3), and Optics has no violation with the test dataset xerces-1.3. NASA datasets KC3 and MW1 have fewer instances than CM1, PC1 and KC1 (Table 3.3), and Optics also has no violation with the test datasets of KC3 and MW1. We found that the size of datasets could affect Optics' sensitivity to MRs, and this is reasonable for Optics, which stores an order list for each object based on its density. Suppose the datasets contain a large number of high-dimensional instances. In that case, it will be hard to create index structures to efficiently support the hypersphere range queries needed by the Optics algorithm [94, 95, 103]. Another potential factor to consider is the shape of the data distribution. Optics tend to cluster data better when the data distribution has a spherical shape, as opposed to other shapes [77]. After performing data transformations with MRs, the shape of the data distribution may change to a form that Optics cannot effectively handle. Therefore, violations of Optics may be caused by both the size of the datasets and the shapes of the data distributions. Further investigations are necessary to identify the root causes related to the data characteristics that lead to violations for Spectral and Optics.

3.5.7.2.2 Caused by configurations of algorithms

Fig. 3.6b shows Affinity violated all MRs with all the datasets, so data should not cause the violations. Although Affinity does not need to set the number of clusters to predict, it needs to set two parameters. They are “damping” and “preference” [104]. “damping” is set to avoid numerical oscillations in the updating process, and the range is from 0.5 to 1. This study sets 0.75 for damping, and the “preference” factor is set to “-1e05”. These numbers made Affinity create two clusters for test datasets of source datasets after a few trials. However, if the follow-up SDPs did not suit these configurations, the predictions of follow-up Affinity would be affected [105, 106]. Therefore, we conclude that the configurations of Affinity cause the violation,

and our generic MT model is unsuitable for Affinity as this is a validation study, and no cross-validation of configurations should be used.

Meanshift shifts each instance iteratively in the dataset until the top of its kernel density estimation surface reaches the nearest peak [77]. However, Meanshift relies on the bandwidth configurations for better clustering results [92]. We used the “estimate_bandwidth” function of the “scikit-learn” library with configurations “quantile=0.5” and “n_sample=500” to get the bandwidth. These configurations did not work with different follow-up datasets, except MR2. Therefore, results show that only MR2 can validate Meanshift without tuning bandwidth configurations. Further investigations are necessary to identify the influence of configuration on Affinity and Meanshift that caused the violation.

3.5.7.2.3 Caused by characteristics of algorithms

Birch treats all instances as an initial cluster at the beginning and then successively splits the cluster into smaller ones until each instance belongs to one cluster or a predefined condition is met [77]. However, Birch is sensitive to the order of the data [81], so Birch will give different clustering results for MR1 (reversing the order of data), MR3 and MR5 (both are adding more data to change the order). In addition, Birch works well with the spherical shape of clusters [81], and MR4 (remove features) can change the shape of data distributions, which affects Birch’s predictions.

MiniKM is insensitive to MR4.1 (Table 3.9). This could be explained by the fact that “mini-batches tend to have lower stochastic noise than individual examples in stochastic gradient descent and allow convergence to better solutions” [107]. However, the characteristics of PBC algorithms are sensitive to noisy data and outliers [81] and randomly select instances as the initial centre points [77]; therefore, it is rational that MR1 (reversing order of data) could affect the selection of the initial centre points for PBC, MR3 (changing the density of data), MR4 (removing features) and MR5 (adding outliers) could make the data noisy to PBC.

The superior performance of BayGM compared to GMM (Table 3.6) can be attributed to BayGM's utilization of smaller batches or truncated distribution, which tends to result in less noise and more accurate solutions [107]. In addition, BayGM(spherical) did not significantly violate MR1 and MR4 (Table 3.10), which could be explained by the shapes of the clusters identified by source BayGM(spherical) and follow-up BayGM(spherical) of MR1 and MR4 are similar in spherical shapes. This similarity in shapes led to consistent prediction outcomes. Conversely, MBC algorithms are sensitive to noisy data and outliers [108]. Since MR3, and MR5 can make data noisy, it is rational that MBC is sensitive to them. In addition, MBC algorithms use the EM algorithm to initialize the model, which obtains an initial estimate of the model parameters randomly [82], and MR1 reverses the order of data can affect the selection of the initial estimates, which makes MBC sensitive to MR1. Thus, MBC algorithms are sensitive to MR1, MR3, and MR5 because of the characteristics of the algorithms.

The average RP of RF were lower than half of the SDPs' average RP, and its average ACC and MCC were among the second highest (Table 3.6). In addition, RF had a high average VR and a low average RP, which shows that RF is moderately sensitive to MRs (Table 3.6). However, RF has several hyper-parameters that influence the performance [109], and as RF ensembles several decision trees and each of them learns from a randomly selected sample of data [99], resulting in prediction inconsistencies. It is because MR1 reverses the order of data, which can affect the selection of samples for RF. MR3, MR4 and MR5 can make the data noisy, which can also affect the selection of samples for RF. Although further tunings for the hyper-parameters of RF can improve the performance of RF, this study prioritizes validating SDPs and the successful tuning of parameters could mask actual errors in program. Thus, due to its characteristics and configurations, RF is sensitive to MR1, MR3, MR4 and MR5.

KNN demonstrated the highest average ACC and MCC, high average VR and low average RP, indicating its moderate sensitivity to MRs (Table 3.6). However, KNN is sensitive to noise in the dataset. Standardization and normalization of the dataset must be implemented for better performance [103, 110]. The inconsistency of KNN could be explained as MR3, MR4, and MR5 can make the data noisy. Furthermore, KNN is sensitive to MR1 because the standardization process involves standardizing the order of data to make KNN stable. KNN also suffers from the influence of “k” hyper-parameter [98]. We chose seven as the “k” value because we used dataset “log4j-1.1”, which is the smallest in size (Table 3.3), to estimate that each cluster should at least have seven neighbours, in order to make KNN works for all the datasets. Although further tunings for parameters can improve the performance of KNN, this study focuses on validating SDPs and excessive tuning of parameters may obscure actual errors in program. Thus, due to its characteristics and configurations, KNN is sensitive to MR1, MR3, MR4, and MR5.

Answering RQ2: After analysing the violated SDPs, we found three underlying causes of violations: characteristics of data, configurations of algorithms and characteristics of algorithms. These causes show the importance of understanding the data and the algorithms before applying the proposed model.

3.5.7.3 RQ3: What are the differences in applying generic MT model to validate USDP and SSDP?

This section discusses the differences in applying MT to validate USDPs and SSDPs. The main difference between them is that SSDP uses labelled datasets, and USDP does not, so this section focuses on the effect of labelled datasets.

3.5.7.3.1 MR2

All SSDPs and seventeen USDPs did not significantly violate MR2 (Tables 3.5 and 3.7), and MR2 shrinks and expands all feature values in datasets proportionally without changing the labels. Thus, the labelling of MR2 follow-up datasets did not affect SSDPs, and this shows that labelled data does not make a difference in applying our MR2 to validate USDPs and SSDPs.

3.5.7.3.2 Correlation between prediction accuracies and inconsistencies

Results show that SSDPs had the highest average ACC and MCC out of all SDPs (Table 3.6), which is expected as SSDPs have labelled data to revise their predictions. Even Agglo and DBscan had zero violation, their average ACC and MCC were lower than SSDPs (Table VI). In addition, Table 3.6 shows that the average RP of SSDPs were lower than thirteen USDPs; however, the average VR of SSDPs were over 70%, similar to other highly violated USDPs. We also ran a correlation analysis with average values of ACC, MCC, VR and RP of all SDPs. Table 3.12 shows that “ACC and MCC” are correlated, and “VR and RP” are correlated; however, “ACC and RP”, and “MCC and RP” are not highly correlated. These results show that prediction inconsistencies, such as VR and RP, are independent of prediction accuracies for SSDPs and USDPs. Thus, labelled datasets can only improve prediction accuracies and make no difference in applying our generic MT model to SSDPs and USDPs, focusing on prediction inconsistencies.

Table 3.12: Correlation analysis based on average values of all SDPs.

	ACC	MCC	VR	RP
ACC	1.00			
MCC	0.72	1.00		
VR	0.43	0.19	1.00	
RP	-0.14	-0.50	0.61	1.00

Answering RQ3: Results show that labelled datasets make no difference when used in our generic MT model for SSDPs and USDPs, as they are insensitive to MR2.

Although SSDPs have higher prediction accuracies, which can be affected by labelled data, the prediction inconsistencies are independent of prediction accuracies.

3.5.7.4 Further Analysis

The results show that although the model is generic because users can apply this model with any numeric data feature studied by machine learning algorithms, Nineteen SDPs did not significantly violate MR2. Thus, MR2 is the generic MR that can serve as a validation tool for any numeric feature of data studied by thirteen machine learning algorithms (except Affinity), even without prior knowledge of the validating systems.

Agglo is the only SDP insensitive to all MRs and can make predictions better than random guesses (over 50%) in the study. Thus, the whole model can be applied as a validation tool for systems that employ the Agglo algorithm, even without prior knowledge of the validating systems.

As the above results show, our generic MT model can be applied without prior knowledge of the validating systems, and our model can be used for continuous monitoring of machine learning based systems. If there are any updates to the machine learning based systems, our model can validate the updated systems. Moreover, if the system violates the MR of our model, errors in the updated systems can be detected at an early stage.

The experiment results did not show any trend from MR3.1 to MR3.7, which did not show any relationship between the density level and the level of re-clustering. Similar to MR5.1 to MR5.7, the results did not show any relationship between the number of outliers and the level of re-clustering. Although four SDPs did not violate MR4.1 and two SDPs did not violate MR4.2, results did not show any relationship between the number of removed features and the level of re-clustering.

3.5.8 Threats to Validity

This section discusses the threats to validity that are faced by this experiment.

3.5.8.1 Internal validity

One internal threat to the experiment is the randomness of clustering systems, as this can cause result variations. This study fixed the random seed for subject SDPs, making the results reproducible in each execution run. Another internal threat to our experiment is related to parameter configuration. Standard parameters, which are mentioned in section 3.5.4, were applied to all SDPs. We built SDPs using the “scikit-learn” library of Python and created the follow-up datasets using Microsoft Excel. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Finally, this study focused on the relationship between MRs and measurements mentioned in section 3.5.5. The prediction outcomes of all SDPs were listed and checked to ensure that the measurements mentioned in section 3.5.5 were calculated accurately.

3.5.8.2 External validity

METTLE and μ MT were proven models [19, 41, 76], and this study is based on these models to develop the generic MT model. The potential threat to the external validity of this study is low. Another external threat is the generality of our model. As we applied DMOs of μ MT [76] and proposed DMOs to the implementation of the proposed model, the DMOs are all generic and can be applied to any numeric datasets. This study has chosen twenty software defect datasets of NASA and PROMISE, which vary in size and features (Table 3.3). Twenty SDPs using fourteen different algorithms were employed. This study ran five trials for each SDP with each MR and dataset (section 3.5.6). The results should be generalised. Although the datasets used for assessment may vary case by case, the subject SDPs assessed and evaluated by

our approach are rather general. Thus, we argue that the effect of different datasets on the effectiveness of the proposed model should be small.

3.5.8.3 Construct validity

This study follows Xie et al.'s RP [19] to measure the effectiveness of our generic MT model. The independent variables of this study are prediction outcomes of the same test datasets of different SDPs, which were trained by different follow-up datasets. The dependent variable is the differences between the independent variables and the prediction outcomes of the same test datasets of SDPs trained by the source dataset, which is the RP. As this study focuses on validating SDPs, RP could show the violations. Suppose there is no violation between source prediction outcomes and follow-up prediction outcomes, which means no inconsistent prediction between them. In that case, RP is zero, and the SDP is insensitive to the related MR. That means the MR could be used for validation. As SDP is combined with programs and trained data, a violation with the MR could reveal an error in programs because the related SDP should be insensitive to the MR. That can correctly measure the intent of this study. To sum up, the construct validity of the independent variables and dependent variables of this article should be acceptable.

3.5.9 Limitations

The proposed model could only apply to features with numeric values, which might not be applicable to the future trend pre-trained language models, and this study only experimented on a two-labelled classification problem and employed two supervised software defect prediction models using machine learning algorithms. Additionally, due to limited computer resources, this study only selected subject datasets of less than 1200 instances.

3.6 Chapter Summary

In this chapter, we conduct an experiment that adapts the existing MT model into a generic MT model for validating ML-based software defect prediction systems. Although USDP is a potential solution for SSDP's insufficient labelled defect training datasets, it is still faced with the test oracle problem. MT is a widely used technique to alleviate the test oracle problem; however, identifying MRs is challenging as these expected relationships are mainly created by experts or users. Thus, this experiment proposed a generic and adaptable MT model to alleviate the challenge of identification of MR. This generic MT model adapted twenty-one generic MRs from five generic types of MRs of METTLE [19] mapping with DMOs of μ MT [41, 76] and proposed DMOs. We experimented with the proposed model with twenty publicly available high-dimensional software defect datasets of NASA and PROMISE, and twenty SDPs implementing fourteen different ML algorithms, in order to answer our research objective, which is to demonstrate the feasibility of the proposed generic MT approach model for validating USDPs using clustering algorithms and SSDPs.

In conclusion, this study achieved the research objective. It proved that the generic MT approach in validation is a promising direction. The proposed generic MT model is a valuable validation tool for SDPs, even with limited knowledge of validating targets and no inputs from experts or users. Firstly, our first research question on whether the generic MT model could validate SDPs and the results showed that it could. Agglo and DBscan did not violate any MR, which indicated that using a "bottom-up" clustering ML approach with no random initial sample selection is highly adaptable and robust to the proposed model. Thus, we recommend using the proposed model for validating SDPs using Agglo and DBscan. In addition, MR "Shrinkage and Expansion" (MR2) is the most generic MR and a necessary property, as nineteen SDPs did not significantly violate this MR. Secondly, our second research question on the cause of the violation and results showed that the causes are

characteristics of data, configurations of algorithms and characteristics of algorithms. Thus, future applications of the proposed model must consider these three factors. Thirdly, our third research question on whether there is a difference in applying the generic MT model to validate USDP and SSDP and results showed that the prediction inconsistencies are independent of prediction accuracies, which means labelled datasets make no difference in using our model to validate SSDPs and USDPs. Fourthly, due to the datasets with less than 1200 instances and less than forty features, the proposed model might not be tested thoroughly because it is not difficult for machine learning systems not to violate MR with such a small number of test instances. Further studies with datasets with much higher features and instances are needed to evaluate the robustness and practicality of the proposed models for future use. Lastly, as the results show, our generic MT model can be applied without prior knowledge of validating ML-based SDPs; our model can be used to monitor ML-based systems continuously.

Although the experiment proved that generic MT can serve as a validation tool for ML-based systems, the proposed model is limited to features with numeric values, which may not be applicable to the pre-trained language models, which are the future trend of ML. The next chapter will focus on uncertainty and use it as the foundational concept to derive generic MRs of MT. We will conduct an experiment to assess uncertainty-based generic MT as a validation tool for ML-based systems.

Chapter 4. Uncertainty-based Metamorphic Testing on Validation of Machine learning-based Systems

The generic Metamorphic Testing (MT) model discussed in Chapter Three is limited to numeric problems. In this chapter, we shift our focus to the role of uncertainty in developing another generic MT approach to address the challenge posed by the limited participation of experts and users, which can hinder the development of effective and suitable Metamorphic Relations (MRs), thereby impacting the validation capacity of MT in the experimental context of Machine Learning-based (ML-based) plagiarism detection systems (PDSs). To tackle the issue of insufficient validation of the performance of ML-based educational systems, we propose using an uncertainty matrix to identify the uncertainty in PDSs and then derive generic MRs of MT for validation.

The rest of this chapter is structured as follows. Section 4.1 introduces the whole work of this chapter. Section 4.2 describes the background of this research under three aspects: plagiarism detection, uncertainty, and MT. Section 4.3 introduces the motivation. Section 4.4 presents the proposed approach. Section 4.5 provides an overview of the evaluation methodology. The results and discussion will be

presented in sections 4.6 and 4.7. We conclude and discuss limitations and future research avenues in section 4.8.

4.1 Introduction

Plagiarism is a serious problem in academia. Despite advanced technologies have been applied in detecting plagiarism to enhance performance [42-45], other advanced technologies have also made detection more difficult [51, 52]. Thus, evaluating the performance of PDSs is essential because incorrect detections will cause disputes [51] and reduce trust in using the systems [1]. The inconsistency in PDSs might lead to uncertainty. It is crucial to evaluate these systems; however, evaluating a PDS is a test oracle problem as it is difficult to distinguish correct behaviour from potentially incorrect behaviour of PDSs. In Chapter 3, the generic MT model has been proven as an effective valuation tool for ML-based systems. However, the proposed model in Chapter 3 can only apply to ML-based systems with numeric features. Thus, this chapter shifts the research objective to demonstrate the feasibility of applying generic MR derived from uncertainty for validating ML-based systems with features other than numbers. We propose an approach by applying an Uncertainty MatriX (UMX) to systematically identify the uncertainty of PDSs and use identified uncertainty to derive generic MRs of MT for validation. This chapter experimented with three different PDSs. We simulated a classroom scenario where students used Artificial Intelligence Generated Content (AIGC) and paraphrasing tools to generate different assignment answers. The generated answers were fed into the subject systems separately for validation by comparing the systems' similarity scores of the tool-generated answers.

We conducted an experiment using three PDSs in a classroom scenario where students were suspected of using tools to generate answers for their assignments. These answers were then inputted into the systems for validation by comparing the similarity scores of the tool-generated answers. The results demonstrated that our

approach can effectively validate PDSs. In the future, this approach could be applied to identify uncertainties and enhance the robustness of ML-based systems.

4.2 Background

This section briefly introduces plagiarism detection using Machine Learning (ML) approaches and uncertainty.

4.2.1 Plagiarism Detection

Plagiarism is often found in academia [112]. Institute of Electrical and Electronics Engineers (IEEE) “defines plagiarism as the reuse of someone else’s prior processes, results, or words without explicitly acknowledging the original author and source”, and IEEE penalised offenders by disallowing publication in all IEEE copyrighted publications for up to five years [113]. Thus, plagiarism is a severe issue. Based on the definition of plagiarism, measuring the similarity of documents is an approach to detecting plagiarism. Several commonly used similarity calculation methods include Manhattan, Euclidean distance, Jaccard coefficients, and Cosine similarity [112, 114-120]. The cosine similarity method is popular because it measures similarity between vectors, which has advantages over others [117].

ML has been applied to improve the performance of detection of text documents such as fraudulent reviews, spam filtering, tag extraction and plagiarism [42-45]. A systematic review study has also found that plagiarism detection was improved by “Better semantic text analysis methods” and “Application of machine learning” [2]. To apply ML techniques to detecting text documents, the ML model typically works with Natural Language Processing (NLP). NLP is one of the computation techniques to capture the semantics of natural language texts and uses different knowledge representations for specific purposes [46-48]. ML models can be based on representations of NLP to make predictions.

Term Frequency Inverse Document Frequency (TFIDF) is a term weighting scheme commonly used in NLP to represent textual documents as vectors [48, 115, 119-121]. TFIDF is widely used and has been applied with cosine similarity method, ML algorithms and word embedding techniques for text classifications, text similarity, and plagiarism detection [114, 116, 119, 120, 122, 123]. Other advanced NLP techniques, such as Sentence-Transformer “SBERT”, “OpenAI Generative Pre-trained Transformer GPT2”, “word2vec”, and “doc2vec”, are also used with cosine similarity method and ML techniques for plagiarism detection, text categorization and noun phrase representation [43, 48, 120-122, 124-126].

Advanced technologies, such as AIGC, have also made detection difficult [51, 52]. No system can detect plagiarism 100% correctly [51], and there was a lack of performance evaluation of PDSs [54, 55]. If the detection is incorrect, there will be severe consequences; thus, the uncertainty of PDSs is of crucial concern [1, 51].

4.2.2 Uncertainty

Uncertainty concerns with the variability of outcomes we are unsure about due to a lack of information, whereas risk refers to the probability of unexpected outcomes. It is necessary to assess the risks that uncertainty imposes on the safety of ML-based systems to increase the credibility of adopting systems [1]. [127]’s study proposed a framework to characterize and document the attributes in uncertainties of systems. [128]’s study introduced the Resampling Uncertainty Estimation algorithm to measure the uncertainty of ML models. [129]’s study proposed an uncertainty matrix based on three dimensions of uncertainty: location, level, and nature, to identify critical uncertainties. After identifying the uncertainty, we need an approach to validate so that the systems can offer performance without uncertainty, and MT is a valuable validating tool.

4.2.3 Metamorphic Testing (MT)

MT is one approach that can alleviate the problem of test oracles [19, 22-24]. MT has been used to validate the expected relationships between inputs and outputs of program executions. These expected relationships are expressed as MRs. If the output of the program executions does not match the expected relationships, this violates the MR and fault is revealed [19, 21, 24, 31]. So, by examining these expected relationships between inputs and outputs, we can validate the program under testing.

4.3 Motivation

From the previous chapter, the proposed generic MT model can serve as a validation tool, although the model can only take on numeric features of ML-based systems. Motivated by the necessity of generic MT, which can take on more than only numeric features, for validating ML-based PDSs that suffer from the test oracle problem and lack of research attention. This experiment aims to develop an uncertainty-based validating approach using MT for PDSs and demonstrate the practicality of the proposed approach through a comprehensive scaled experiment.

4.4 Proposed Approach

This section describes the validation model and how the Uncertainty-Based generic MT approach is designed.

4.4.1 Uncertainty-Based Metamorphic Testing

We adopted [129]’s uncertainty matrix (Table 4.1) to identify the uncertainty of the PDSs. One of the “Scenario” level of uncertainties of PDSs is inconsistent detection, and the cause of this uncertainty can be located in “Context”, “Model Structure”, and “Input” (Table 4.1). This study focuses on the “Input” location because the subject

PDSs were using default options and experimented in the same test scenario (section 3), so we assumed “Context” and “Model Structure” were constant. Thus, we identified that the uncertainty of PDSs is due to having a range of values of inputs (Table 4.1).

After identifying the cause of uncertainty, we adopted the MT approach to validate PDSs. We derive the MR of MT based on the identified cause and create MR called “MR1: Paraphrasing object”. This relation assumes that the similarity of the document should not be affected by paraphrasing. In other words, systems should identify the “Source” document and the “Follow-up” documents, which are created by MR1, as similar (Fig. 4.1). MR1 is the relation that should be held. If not, there is a violation with MR1 and uncertainty in the PDS.

Table 4.1: Alwidian et al. [129]’s uncertainty matrix

Location	Level			Nature	
	Statistical	Scenario	Recognized Ignorance	Epistemic	Variability
Context	deviation from true values	External / future environment	Possible at any location	Possible at any location	Possible at any location
Model Structure	deviation from true values	different assumption about structure			
Input	measurement uncertainty	Range of values			
Model Outcome	accumulated uncertainty which results from uncertainty in location X level X nature				

Source doc = [(man eats food)] Follow-up doc = [(male human consumes edibles)] ← paraphrased by MR1 Therefore, System predicts (Source doc, Follow-up doc) = similar

Figure 4.1: Illustration of MR1: Paraphrasing object

4.5 Experimental with uncertainty-based metamorphic testing on validation of machine learning-based plagiarism detection systems

This section describes the setup of the experiment. It defines the research objective and research questions of our experiment. Then, it describes the subject systems and data used in the experiment. Thereafter, it discusses detailed experimental design, including environment configuration, measurement, experimental procedure, dataset preparation, and parameter setting. At last, we present our experimental results and further discuss, analyse and interpret the results for research questions.

4.5.1 Research Questions

The main objective of this experiment is to attempt to answer the research question.

RQ1: Can we validate machine learning-based Plagiarism detection systems by generic MR of MT derived from the uncertainty matrix?

We derived the MR from the uncertainty of PDSs and evaluated the effectiveness of the MR in validating subject systems by counting the occurrence of violations related to MR.

4.5.2 Experiment scenario

Inspired by [130]’s study, this experiment simulates a “single assignment question multiple assignment answers” classroom scenario. Three different assignment questions were given to three classes of twelve students. One Computer Science (CS) assignment was expected to return Java source codes. Two Law assignments (Law1 and Law2) were open questions and expected to return English text documents.

4.5.3 Subject systems and technologies under test (SUT)

4.5.3.1 Plagiarism detection system (PDS)

SBERT and TFIDF are commonly used word embedding techniques for plagiarism detection [120, 121], so this study employed them. We focused on PDSs that measure the similarity between two documents, and three subject PDSs were set up.

Inspired by [118], the first PDS, **BERT_COS**, using SBERT Transformer with the pre-trained model “bert-base-uncased” and cosine similarity method.

The second PDS, **NN-TFIDF**, follows [131]’s unsupervised implementation using TFIDF with cosine similarity method and Nearest Neighbour algorithm.

The third PDS, **Neu-SBERT**, follows [132]’s implementation using SBERT Transformers, neural network algorithm and Stanford Natural Language Inference Corpus (SNLI). SNLI is a supervised dataset for sentence similarity with three labels: “contradiction” means the sentences share no similarity, “entailment” means the sentences have a similar meaning, and “neutral” means the sentences are neutral. Neu_SBERT is trained with SNLI corpus and derived semantically meaningful sentence embeddings for similarity prediction with the labels.

Artificial Intelligence Generated Content (AIGC). AIGC refers to using Artificial Intelligence (AI) to generate different kinds of content, including text, images, videos and audio [133, 134]. “ChatGPT”, one of the AIGC applications that can provide detailed responses based on users’ input [134], had been discussed its relationship with plagiarism [51, 52]. This study employed two different AIGC applications from “poe.com”: **ChatGPT**, powered by OpenAI’s GPT3.5 technology, and **Claude**, powered by Anthropic’s technology [135]. We assumed each student would use one of the two AIGC applications to generate assignment answers.

4.5.3.2 Paraphrasing tools

Online paraphrasing tools could make detection more difficult for PDSs [136], and we selected five different free online paraphrasing tools for MR1 (Table 4.2). In this

study, each student would use zero or one of the paraphrasing tools to paraphrase the generated answers.

Table 4.2: List of online free paraphrasing tools

ID	Site name	Web address
p1	Paraphrasing-Tool	https://paraphrasing-tool.com/
p2	Quillbot	https://quillbot.com/
p3	PrePostSeo	https://www.prepostseo.com/paraphrasing-tool
p4	Rewriter-tools	https://www.rewritertools.com/paraphrasing-tool
p5	SEOMagnifier	https://seomagnifier.com/online-paraphrasing-tool

4.5.3.3 Explainable Artificial Intelligence (XAI)

We selected Shapley XAI technology for internal validity. **SHAP** measures each feature’s contribution towards the prediction [137]. It is based on coalitional game theory, which explains prediction by “assuming that each feature value of the instance is a player in a game where the prediction is the payout”, and the Shapley values are calculated this way and explain how to distribute the prediction among the features [138].

4.5.4 Configuration

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “scikit_learn” and “tensorflow”, for machine learning algorithms, “shap” is for XAI, and “transformer” is for language processing. Standard parameters and options were applied to SUTs.

4.5.5 Subject Data

According to MR1, original and paraphrased questions or paraphrased and original answers should give similar results. For the CS assignment, as source codes might not be paraphrased because paraphrasing might lead to the source codes being uncompileable. Instead, we assumed students would paraphrase the CS assignment question to create slightly different answers to avoid plagiarism detection. So, we paraphrased the CS questions to create follow-up questions and then fed them into AIGC tools to generate CS answers. Including the model answer of CS, the total number of CS answers was thirteen (Table 4.3). For Law assignments, we fed the original question into AIGC tools to generate original LAW1 and LAW2 answers. Then, we use paraphrasing tools to generate follow-up answers. Including the model answers of LAW1 and LAW2, the total number of Law1 and Law2 answers was twenty-six. Therefore, this experiment had thirty-nine answers (data) (Table 4.3).

Table 4.3: Details of subject data

Type of assignment	CS	LAW1	LAW2
<u>Model answer</u>			
- original	std_answer-cs	std_answer-law1	std_answer-law2
<u>Generated answer</u>			
By ChatGPT			
- original	gpt-org-cs	gpt-org-law1	gpt-org-law2
- paraphrased by p1	gpt-p1-cs	gpt-p1-law1	gpt-p1-law2
- paraphrased by p2	gpt-p2-cs	gpt-p2-law1	gpt-p2-law2
- paraphrased by p3	gpt-p3-cs	gpt-p3-law1	gpt-p3-law2
- paraphrased by p4	gpt-p4-cs	gpt-p4-law1	gpt-p4-law2
- paraphrased by p5	gpt-p5-cs	gpt-p5-law1	gpt-p5-law2
<u>Generated answer</u>			
By Claude			
- original	cla-org-cs	cla-org-law1	cla-org-law2
- paraphrased by p1	cla-p1-cs	cla-p1-law1	cla-p1-law2
- paraphrased by p2	cla-p2-cs	cla-p2-law1	cla-p2-law2
- paraphrased by p3	cla-p3-cs	cla-p3-law1	cla-p3-law2
- paraphrased by p4	cla-p4-cs	cla-p4-law1	cla-p4-law2
- paraphrased by p5	cla-p5-cs	cla-p5-law1	cla-p5-law2
Total	13	13	13

4.5.6 Measurement

This study measures the correctness of plagiarism detections, so subject PDSs should identify which answers were generated by the same AIGC application, even after paraphrasing. We used the similarity score calculated by PDSs to rank the similarity of answers. If PDS identifies a ChatGPT-generated answer more similar to a Claude-generated answer, this would be counted as a violation as the paraphrased answer misled PDS (Table 4.4). The violation rate will be measured by the violations over the total number of comparisons, which is one hundred and thirty-two as Equation (4.1).

$$\text{Violation rate} = \text{Violation count} / \text{Total number of comparisons} \quad (4.1)$$

Table 4.4: Illustration of violating MR1

Answer	Similarity score with "gpt-org-law1"	Compare
cla-p5-law1	0.95	
gpt-p1-law1	0.91	Violation
gpt-p2-law1	0.96	
gpt-p3-law1	0.96	
gpt-p4-law1	0.96	
gpt-p5-law1	0.96	

4.5.7 Experiment implementation

Step 1. For CS, an original question and five paraphrased questions, which were paraphrased by tools listed in Table 4.2 as MR1 stated, are inputted into two AIGC tools to generate answers. Answers from the original and paraphrased questions are marked in Table 4.3. For LAW1 and LAW2, original questions were inputted into two AIGC tools to generate original answers, and then the original answers were paraphrased by tools (Table 4.2) to create paraphrased answers (Table 4.3).

Step 2. We considered the generated answers without paraphrasing as the “Source” answers and the paraphrased answers as the “Follow-up” answers. Therefore, MR1 expects “Source” answers to be similar to “Follow-up” answers. CS and Law assignment answers were fed into PDSs separately. The results of similarity scores

were used to compare and rank (Fig. 4.2) for counting violation of MR1 (section 4.5.6).

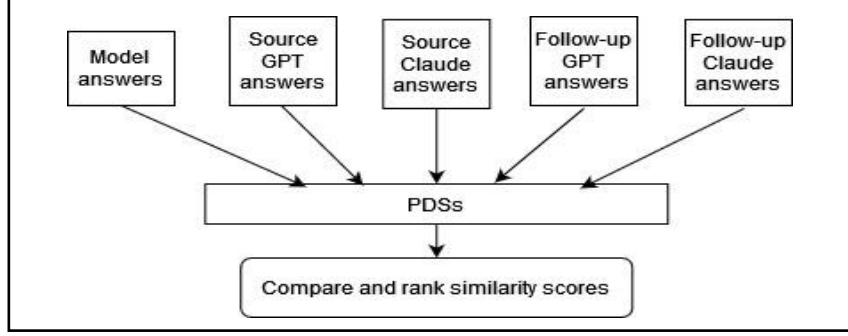


Figure 4.2: Implementation of experiment.

4.5.8 Experimental Results and Discussion

We first studied zero violation of MR with all subject PDSs to evaluate the effectiveness of our proposed approach (RQ1). Then, we further analysed the violations to seek more understanding of the causes.

4.5.8.1 Zero violation

Table 4.5 shows that all PDSs performed better in CS than LAW1 and LAW2. BERT_COS did not violate MR1 with CS answers, which means BERT_COS could identify all the CS answers generated by the same AIGC tool as most similar to each other (Table 4.6). NN_TFIDF and Neu_SBERT did not violate MR1 with CS answers generated by Claude (Table 4.5, 4.7 and 4.8). It is because CS answers were Java source codes that were similar in size and highly structural for compilation.

Table 4.5: Violation counts in the experiment

	CS			LAW1			LAW2		
	BERT_COS	NN_TFIDF	Neu_SBERT	BERT_COS	NN_TFIDF	Neu_SBERT	BERT_COS	NN_TFIDF	Neu_SBERT
Violation(gpt)	0	6	6	3	8	6	2	6	13
Violation(cla)	0	0	0	1	8	9	4	2	6
Total violation	0	6	6	4	16	15	6	8	19
Total comparison	132	132	132	132	132	132	132	132	132
Violation rate	0%	4.55%	4.55%	3.03%	12.12%	11.36%	4.55%	6.06%	14.39%

Table 4.6: Matrix of similarity scores of BERT_COS with CS

CS -assignment-sim-score													
	std_answer-cs	cla-org-cs	gpt-org-cs	cla-p1-cs	cla-p2-cs	cla-p3-cs	cla-p4-cs	cla-p5-cs	gpt-p1-cs	gpt-p2-cs	gpt-p3-cs	gpt-p4-cs	gpt-p5-cs
std_answer-cs		0.9598	0.9872	0.9598	0.9598	0.9598	0.9598	0.9598	0.9872	0.9872	0.9872	0.9872	0.9872
cla-org-cs	0.9598		0.9264	1.0000	1.0000	1.0000	1.0000	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264
gpt-org-cs	0.9872	0.9264		0.9264	0.9264	0.9264	0.9264	0.9264	1.0000	1.0000	1.0000	1.0000	1.0000
cla-p1-cs	0.9598	1.0000	0.9264		1.0000	1.0000	1.0000	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264
cla-p2-cs	0.9598	1.0000	0.9264	1.0000		1.0000	1.0000	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264
cla-p3-cs	0.9598	1.0000	0.9264	1.0000	1.0000		1.0000	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264
cla-p4-cs	0.9598	1.0000	0.9264	1.0000	1.0000	1.0000		1.0000	0.9264	0.9264	0.9264	0.9264	0.9264
cla-p5-cs	0.9598	1.0000	0.9264	1.0000	1.0000	1.0000	1.0000		0.9264	0.9264	0.9264	0.9264	0.9264
gpt-p1-cs	0.9872	0.9264	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264		1.0000	1.0000	1.0000	1.0000
gpt-p2-cs	0.9872	0.9264	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264	1.0000		1.0000	1.0000	1.0000
gpt-p3-cs	0.9872	0.9264	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264	1.0000	1.0000		1.0000	1.0000
gpt-p4-cs	0.9872	0.9264	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264	1.0000	1.0000	1.0000		1.0000
gpt-p5-cs	0.9872	0.9264	1.0000	0.9264	0.9264	0.9264	0.9264	0.9264	1.0000	1.0000	1.0000	1.0000	

Answering RQ1: MR1 can validate BERT_COS with CS answers generated by both AIGC tools and can validate NN_TFIDF and Neu_SBERT with CS answers generated by Claude.

Table 4.7: Matrix of similarity scores of NN_TFIDF with CS

CS -assignment-sim-score													
	std_answer-cs	cla-org-cs	gpt-org-cs	cla-p1-cs	cla-p2-cs	cla-p3-cs	cla-p4-cs	cla-p5-cs	gpt-p1-cs	gpt-p2-cs	gpt-p3-cs	gpt-p4-cs	gpt-p5-cs
std_answer-cs		0.7657	0.9650	0.7705	0.7728	0.7728	0.7728	0.7728	0.8542	0.9682	0.9833	0.9833	0.9833
cla-org-cs	0.7657		0.7308	0.9937	0.9931	0.9931	0.9931	0.9931	0.5898	0.6969	0.7092	0.7092	0.7092
gpt-org-cs	0.9650	0.7308		0.7354	0.7372	0.7372	0.7372	0.7372	0.8313	0.9458	0.9810	0.9810	0.9810
cla-p1-cs	0.7705	0.9937	0.7354		0.9993	0.9993	0.9993	0.9993	0.5935	0.7012	0.7137	0.7137	0.7137
cla-p2-cs	0.7728	0.9931	0.7372	0.9993		1.0000	1.0000	1.0000	0.5946	0.7028	0.7152	0.7152	0.7152
cla-p3-cs	0.7728	0.9931	0.7372	0.9993	1.0000		1.0000	1.0000	0.5946	0.7028	0.7152	0.7152	0.7152
cla-p4-cs	0.7728	0.9931	0.7372	0.9993	1.0000	1.0000		1.0000	0.5946	0.7028	0.7152	0.7152	0.7152
cla-p5-cs	0.7728	0.9931	0.7372	0.9993	1.0000	1.0000	1.0000		0.5946	0.7028	0.7152	0.7152	0.7152
gpt-p1-cs	0.8542	0.5898	0.8313	0.5935	0.5946	0.5946	0.5946	0.5946		0.8823	0.8552	0.8552	0.8552
gpt-p2-cs	0.9682	0.6969	0.9458	0.7012	0.7028	0.7028	0.7028	0.7028	0.8823		0.9659	0.9659	0.9659
gpt-p3-cs	0.9833	0.7092	0.9810	0.7137	0.7152	0.7152	0.7152	0.7152	0.8552	0.9659		1.0000	1.0000
gpt-p4-cs	0.9833	0.7092	0.9810	0.7137	0.7152	0.7152	0.7152	0.7152	0.8552	0.9659	1.0000		1.0000
gpt-p5-cs	0.9833	0.7092	0.9810	0.7137	0.7152	0.7152	0.7152	0.7152	0.8552	0.9659	1.0000	1.0000	

Table 4.8: Matrix of similarity scores of Neu_SBERT with CS

CS -assignment-sim-score (score of entailments)													
	std_answer-cs	cla-org-cs	gpt-org-cs	cla-p1-cs	cla-p2-cs	cla-p3-cs	cla-p4-cs	cla-p5-cs	gpt-p1-cs	gpt-p2-cs	gpt-p3-cs	gpt-p4-cs	gpt-p5-cs
std_answer-cs		0.2181	0.6635	0.2198	0.2181	0.2181	0.2181	0.2181	0.2302	0.2302	0.2302	0.2302	0.2302
cla-org-cs	0.3659		0.3911	0.5859	0.6009	0.6009	0.6009	0.6009	0.2511	0.2511	0.2511	0.2511	0.2511
gpt-org-cs	0.4554	0.2508		0.2475	0.2508	0.2508	0.2508	0.2508	0.2354	0.2354	0.2354	0.2354	0.2354
cla-p1-cs	0.3674	0.5997	0.3872		0.5997	0.5997	0.5997	0.5997	0.2522	0.2522	0.2522	0.2522	0.2522
cla-p2-cs	0.3659	0.6009	0.3911	0.5859		0.6009	0.6009	0.6009	0.2511	0.2511	0.2511	0.2511	0.2511
cla-p3-cs	0.3659	0.6009	0.3911	0.5859	0.6009		0.6009	0.6009	0.2511	0.2511	0.2511	0.2511	0.2511
cla-p4-cs	0.3659	0.6009	0.3911	0.5859	0.6009	0.6009		0.6009	0.2511	0.2511	0.2511	0.2511	0.2511
cla-p5-cs	0.3659	0.6009	0.3911	0.5859	0.6009	0.6009	0.6009		0.2511	0.2511	0.2511	0.2511	0.2511
gpt-p1-cs	0.3772	0.4606	0.4544	0.4486	0.4606	0.4606	0.4606	0.4606		0.6669	0.6669	0.6669	0.6669
gpt-p2-cs	0.3772	0.4606	0.4544	0.4486	0.4606	0.4606	0.4606	0.4606	0.6669		0.6669	0.6669	0.6669
gpt-p3-cs	0.3772	0.4606	0.4544	0.4486	0.4606	0.4606	0.4606	0.4606	0.6669	0.6669		0.6669	0.6669
gpt-p4-cs	0.3772	0.4606	0.4544	0.4486	0.4606	0.4606	0.4606	0.4606	0.6669	0.6669	0.6669		0.6669
gpt-p5-cs	0.3772	0.4606	0.4544	0.4486	0.4606	0.4606	0.4606	0.4606	0.6669	0.6669	0.6669	0.6669	

4.5.8.2 Violation with MR1 and BERT_COS

BERT_COS had the lowest violation rate with Law assignment answers (Table 4.5). LAW1 and LAW2 questions were analysed type with no specific structure for answers. The low violation rate represented that SBERT with the pre-trained model had built a semantic relationship that could identify most LAW answers generated by the same AIGC tool as most similar to each other, even after paraphrasing. In addition, Claude generated LAW1 and LAW2 answers, which were about 330 words and 280 words, respectively. On the other hand, ChatGPT generated LAW1 and LAW 2 answers, which were about 700 words and 440 words, accordingly. The size differences in LAW1 answers were significant between ChatGPT-generated and Claude-generated, which made it easier to distinguish. However, BERT_COS performed only 1.55% better in LAW1 than in LAW2 (Table 4.5), so the size difference effect was insignificant. XAI could not be applied in BERT_COS because it did not use a machine learning technique.

4.5.8.3 Violation with MR1 and NN_TFIDF

Table 4.7 shows NN_TFIDF identified some ChatGPT-generated CS answers were more similar to model answer, such as gpt-org-cs had a higher similarity score with std_answer-cs (0.9650) than with gpt-p1-cs (0.8313). As NN_TFIDF uses term frequencies to measure similarity and CS answers were highly structural and short, NN_TFIDF is expected to perform as well as BERT_COS.

So, we used SHAP to interpret one of the violated cases of NN_TFIDF. Fig. 4.3(a) and (b) show that “particle” was a negative feature of std_answer-cs and gpt-org-cs, whereas Fig. 4.3(c) shows “particle” was a positive feature of gpt-p1-cs. They explained the violation of NN-TFIDF in CS and showed that ChatGPT could generate source codes that made NN_TFIDF difficult to detect plagiarism.

Despite NN_TFIDF violated MR1 with LAW1 and LAW2, NN_TFIDF performed better in LAW2 (Table 4.5). LAW1 answers are longer and paraphrased,

which might affect the term frequencies, leading to increase violations. SHAP interpreted one of the violated cases in NN_TFIDF with LAW1. Fig. 4.4(a) and (c) show that “international” was a positive feature of cla-org-law1 and gpt-p1-law, whereas “international” was a negative feature of gpt-org-law1.

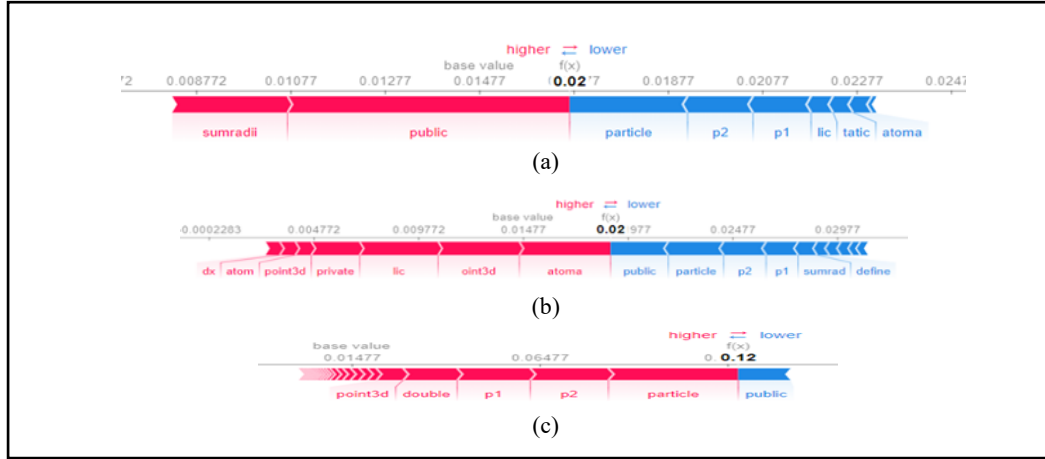


Figure 4.3: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)std_answer-cs, (b)gpt-org-cs, (c)gpt-p1-cs.

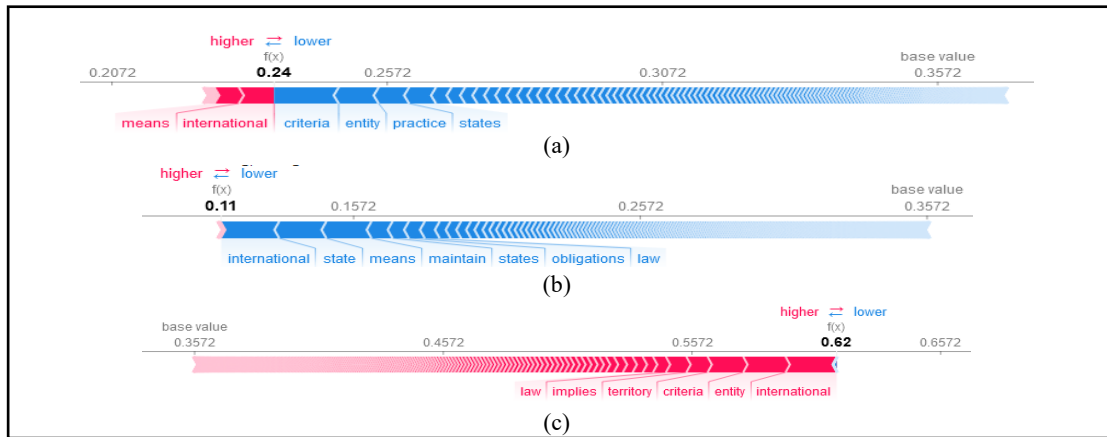


Figure 4.4: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)cla-org-law1, (b)gpt-org-law1, (c)gpt-p1-law1.

In addition, SHAP interpreted one violated case in LAW2. Fig. 4.5(a) and (b) show “international” was a negative feature of cla-p3-law2 and gpt-p2-law2; however, a positive feature of gpt-p5-law2. Apart from explaining the violations, Fig. 4.5(a), (b) and (c) had more common features than Fig. 4.4(a), (b) and (c), which can explain

why NN_TFIDF performed worse in LAW1 than LAW2. It is because LAW1 answers, which were longer and paraphrased, contained more different terms which affected the term frequencies of NN_TFIDF to detect plagiarism.

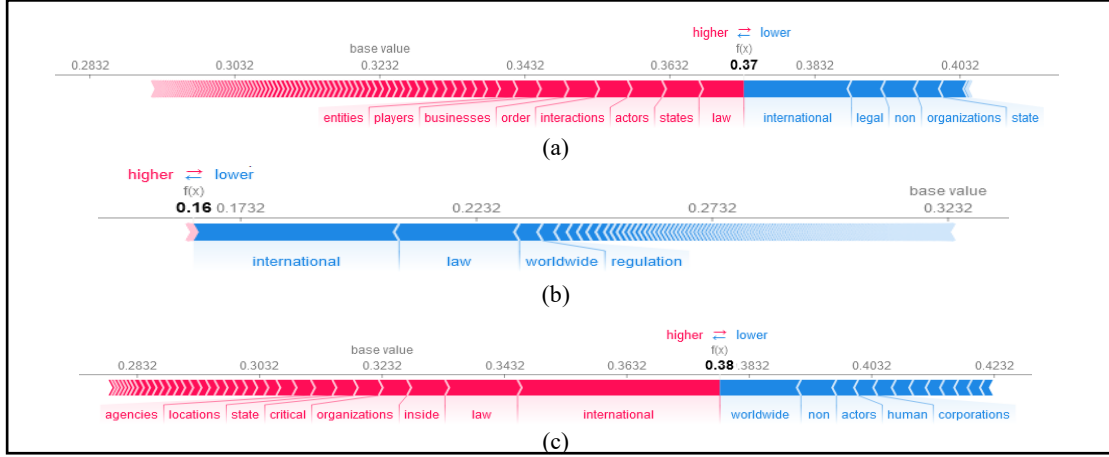


Figure 4.5: SHAP interpretation on features that influenced NN_TFIDF's similarity prediction with (a)gpt-p2-law2, (b)cla-p3-law2, (c)gpt-p5-law2.

4.5.8.4 Violation with MR1 and Neu_SBERT

Table 4.8 shows Neu_SBERT identified some ChatGPT-generated CS answers to be more similar to model answer and Claude-generated answers, such as all five paraphrased ChatGPT-generated CS answers had a higher similar score with cla-p1-cs (0.2522) than with gpt-org-cs (0.2354). Neu_SBERT also used SBERT transformer as BERT_COS and CS answers were highly structural and short, Neu_SBERT is expected to perform as well as BERT_COS. We used SHAP to interpret one of the violated cases of Neu_SBERT and found that the term “public double x” is a negative feature of gpt-p1-cs and gpt-org-cs, whereas cla-p1-cs did not have “public double x”. It made Neu_SBERT identify gpt-p1-cs was more similar to cla-p1-cs than gpt-org-cs. The interpretations explained Neu_SBERT's violations and showed that its semantic relationship to detect similarities between

ChatGPT-generated CS answers was inaccurate. Thus, ChatGPT could generate source codes that made Neu_SBERT challenging to detect plagiarism.

Unexpectedly, Neu_SBERT performed poorly with both Law assignment answers (Table 4.5), even though it could build semantic relationships between words and sentences to detect similarities. We used SHAP to interpret one of the violated cases in LAW1 and found the features that contribute to the decisions of “entailment” of cla-p5-law1, gpt-p1-law1 and cla-org-law1. The term “that decide whether” was a negative feature of cla-p5-law1. cla-org-law1 contains the term “that determine whether”, which is semantically similar to the term “that decide whether”. On the other hand, gpt-p1-law1 does not contain any term similar to “that decide whether”, which made Neu_SBERT identify cla-p5-law1 was more similar to gpt-p1-law1 than cla-org-law1. The interpretations explained Neu_SBERT’s violations and showed that Neu_SBERT could not build an accurate semantic relationship with legal contents for predicting similarities.

4.5.8.5 Further Analysis

The experiment showed that using the uncertainty to derive MR is useful as this approach can identify the core concern of PDSs. If PDS do not violate this MR, the PDS is considered reliable. As BERT_COS is insensitive to MR1 with CS, we can assure the performance of BERT_COS in detecting plagiarism of CS.

Understandably, BERT_COS outperformed NN_TFIDF because SBERT, with a pre-trained model, can build semantic relationships between words and sentences, which is more suitable in LAW1 and LAW2. However, even though CS answers were highly structural and shorter than Law answers, results show BERT_COS outperformed NN_TFIDF in CS, which shows SBERT with a pre-trained model can also understand source code structure and outperform TFIDF.

Additionally, the LAW2 answer is longer, requiring more words. It is reasonable to conclude that NN_TFIDF performed better, as a larger word count increases the

likelihood of detecting plagiarism through word frequencies. However, the increased number of words also makes it more challenging for “semantic understanding” LLMs, as they need to establish more connections. Therefore, NN_TFIDF is likely to be more effective than Neu_SBERT.

At last, BERT_COS outperformed Neu_SBERT in the study, even though they used an SBERT transformer. The critical difference is the training corpus. While BERT_COS uses a pre-trained model, Neu_SBERT uses the SNLI corpus. SNLI corpus could have contained fewer source codes and legal sentences for training than the pre-trained model. Thus, PDSs should choose a pre-trained model over SNLI corpus in detecting plagiarism of CS and LAW assignment answers.

4.5.9 Validity

4.5.9.1 Internal validity

Different input parameters would lead to totally different results. In this experiment, Standard parameters and options (Section 4.5.4) were applied to all subject technologies and systems. We build subject systems using Python libraries and five online tools (Section 4.5.3 and 4.5.4) to create the "follow-up" datasets. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Finally, this study focused on the relationship between MR and measurements mentioned in section 4.4 and 4.5.6. The outcomes were listed and checked to ensure that the measurements mentioned in Section 4.5.6 were calculated accurately. In addition, XAI was applied to interpret violated cases, and the explanations matched the results. Thus, the potential threat to the internal validity of this study is low.

4.5.9.2 External validity

MT was widely used, and this study is based on a generic MT approach. We applied the approach to systems that were used in other published studies. Although the

datasets used for assessment may vary case by case, the subject systems assessed and evaluated by our approach are rather general. Thus, the potential threat to the external validity of this study is also low.

4.5.10 Limitations

This experiment has encountered several limitations. Firstly, we did not include human-created answers in the study. Secondly, this study focuses on the similarity of two documents, and to the best of our knowledge, the SNLI corpus was the only supervised dataset for sentence and document similarity. Thirdly, the proposed approach is based on uncertainties, which might still require experts or users' inputs to identify uncertainty, even with the matrix, thus, this approach might be not generic enough.

4.6 Chapter summary

In this chapter, we conduct an experiment that derives generic MT from uncertainty for validating ML-based PDSs. The reliability of PDSs is essential; however, evaluating a PDS is a test oracle problem as it is challenging to distinguish correct behaviour from potentially incorrect behaviour of PDSs. Thus, this study proposed an approach that applies an uncertainty matrix to systematically identify the uncertainty of PDSs and uses the identified uncertainty to derive the MR of MT for validating PDSs.

We experimented with the approach with three different PDSs. We used AIGC and paraphrasing tools, as MR stated, to generate different answers for validation by comparing the systems' similarity scores of the answers. Results show BERT_COS did not violate MR1 with all CS answers, and NN_TFIDF and Neu_SBERT did not violate MR1 with CS answers generated by Claude. It is because CS answers were well-structured source code programs and were similar in size. Results also show that PDS using SBERT with a pre-trained model outperformed PDS using TFIDF

and SBERT trained by SNLI corpus. It is because the semantic relationship built by BERT_COS with a pre-trained model can understand source code and legal-related sentences better than the other two PDSs. In conclusion, this study proved that the proposed approach can systematically and effectively identify uncertainty and derive associated MR for validating PDSs. We recommend using the proposed approach for validating PDSs using SBERT with a pre-trained model and cosine similarity method to detect plagiarism in highly structural and similar-in-size documents.

The result proved that uncertainty-based generic MT can serve as a validation tool for ML-based systems and incorporate more than just numeric features. This fulfils the research objective of demonstrating the feasibility of applying generic MR derived from uncertainty for validating ML-based systems with features other than numbers. However, the proposed approach relies on uncertainties, which may require input from experts or users to identify, suggesting it might not be entirely generic, even with the use of the matrix. Therefore, the next chapter will shift the focus to metamorphic relation patterns and conduct an experiment to evaluate the generic MRs of MT derived from these patterns to validate ML-based systems.

Chapter 5. Metaphoric Relation Patterns of Metamorphic Testing for Validating Machine learning-based Systems

The previous chapter showed that uncertainty-based generic metamorphic testing (MT) is a valuable validation tool for machine learning-based (ML-based) systems. However, input from experts and users may be necessary to identify sources of uncertainty. This chapter focuses on utilizing metamorphic relation patterns (MRPs) to identify generic metamorphic relations (MRs) for MT. This approach aims to address the challenge posed by the limited participation of experts and users, which can hinder the development of effective and appropriate MRs and, consequently, affect the validation capacity of MT.

The rest of this chapter is structured as follows. Section 5.1 introduces the whole work of this chapter. Section 5.2 describes the background of this research under four aspects: plagiarism, similarity, and plagiarism detection. Section 5.3 introduces the motivation. Section 5.4 presents the proposed MRP. Section 5.5 provides an overview of the experiment, including the experiment's design, the experiment's

implementation process, the results, the discussion, the threats, the conclusion and future directions. We conclude this chapter in section 5.6.

5.1 Introduction

Plagiarism is a serious misconduct in the academic world, and there are detection systems, such as “Turnitin”, to help in finding plagiarism violations. Therefore, the performance and correctness of these tools are critical. In Chapter 4, we proved that uncertainty-based generic MT can serve as a validation tool for ML-based plagiarism detection systems (PDS). However, it might not be entirely generic, even with the use of the matrix. Thus, the research objective of this chapter is to demonstrate that generic MRs of MT, which are derived from MRP, can validate PDS. We proposed three MRPs and experimented on four query-based PDSs that use four different and commonly used embedding techniques to demonstrate the capacity of generic MT. These generic MRPs modify the query contents with the aim of preserving the characteristics of source test outputs. They generate follow-up test inputs while the characteristics of the source test outputs are preserved. If the proposed MRPs are violated, faults in the system will be detected. We conducted the experiment using four PDSs to address the gap in research related to validating the performance of ML-based educational systems. The results show that generic MT, utilising MRPs, can systematically and effectively validate ML-based PDSs, representing a significant advancement in integrated educational technology.

5.2 Background

This section briefly introduces similarity measurements and plagiarism detection using machine learning approaches.

5.2.1 Plagiarism

We have covered the definition of Plagiarism in Chapter 4.2.1. Thus, committing plagiarism can bring serious consequences.

5.2.2 Similarity

To detect plagiarism, it is necessary to have methods measuring similarity. Several common similarity calculation methods include Manhattan, Euclidean distance, and Jaccard coefficients [115, 118, 120], but Cosine similarity is widely used [112, 114-117, 119] because it measures similarity between vectors which has natural advantages over other methods such as Euclidean distance, especially as the dimensionality increases, the characterization ability of similarity based on Euclidean distance cannot be guaranteed [117].

5.2.3 Plagiarism Detection with machine learning

Foltynek et al. [2] reviewed 239 research papers related to plagiarism from the year 2013 to the year 2018. They found that plagiarism detection was improved by “Better semantic text analysis methods” and “Application of machine learning” [2]. Natural Language Processing (NLP) is one of the approaches for detecting document similarity that captures the semantics of a sequence of words, understands the natural language texts and uses various knowledge representations for specific purposes [46-48].

Term frequency–inverse document frequency (tf-idf) is a term weighting scheme commonly used in NLP to represent textual documents as vectors, and it is one of the approaches in the Vector Space Model (VSM) [48, 115, 119-121]. tf-idf has been applied with cosine similarity measures for text similarity detection [114, 116, 119], applied with K Nearest Neighbour (KNN) machine learning algorithm and cosine similarity measures for text classifications [120], combining with word embedding

network for plagiarism detection [122], and combining with latent semantic analysis (LSA) and cosine similarity to detect plagiarism in source codes [123].

Apart from tf-idf, a growing number of papers use advanced word embedding techniques for plagiarism detection or text categorization [48, 124]. For example, KNN has been applied with “word2vec” word embedding method and cosine similarity measures were applied in categorizing texts in news articles and opinions [120], a combination of different word embedding techniques (“word2vec”, “doc2vec”) with varying methods of calculation (sentence vector calculation, cosine similarity or tf-idf weighting technique) to perform plagiarism detection [43, 122, 126], applying different word embedding techniques (“doc2vec”, “SBERT”, “InferSent”, and “Universal Sentence Encoder”) for noun phrase representation [124], etc.

Performance evaluations of PDSs are a test oracle problem since these ML-based PDSs deliver a local optimum rather than a global optimum. A feasible approach for validating these systems is needed, and MT can be used to alleviate the problem of test oracles [19, 22-24]. Zhou et al. [24] further the concept of MRs and defined MRPs and input pattern that can be used to derive multiple concrete MRs. Segura et al. [31] further applied the MR concept and presented a list of MRPs to assist in identifying MRs in testing query-based systems.

5.3 Motivation

Firstly, although advanced technologies have improved plagiarism detection, [54]’s research on 15 web-based text-matching systems that can be used when plagiarism is suspected found “they are still incomplete for guiding the selection of a tool, as they do not evaluate and compare the performance of software systems and their usability from the viewpoint of end-users” [54]. Another two literature studies, which reviewed more than 300 research papers on plagiarism, found a research gap in the lack of methodologically thorough performance evaluations of PDSs [2, 55].

Thus, the literature showed a need to raise awareness of the need to research the assessment of PDSs and plagiarism detection tools.

Secondly, MT is useful in validating ML-based systems. However, the MRs are created mainly through users or experts, which might not be available and might introduce human bias. Thus, we aim to use a generic approach based on [31]’s MRPs for MR creations to remove relying on users and experts and reduce the heuristic influence on MR creation.

Therefore, we aim to identify different generic MRs by MRPs and demonstrate generic MT’s practicality in validating ML-based PDSs through a comprehensive scaled experiment.

5.4 Proposed Approach

In this section, we presented the proposed MRP to use, the experiment's design, and the experiment's implementation.

5.4.1 Generic Metamorphic Relations Pattern (MRP)

In this experiment, we preliminary select the three MRPs from Segura et al.’s [31] list, and they are as follows;

Shuffling. “source and follow-up outputs should contain the same items regardless of the ordering criteria specified as input”. They should give the same outputs.

Input equivalence. “source and the follow-up test cases are equivalent and therefore their outputs should contain the same items in the same order”, e.g., 1 meter = 100 centimetres. They should give the same outputs.

Partition difference. The outputs of the follow-up test cases, which are pairwise disjoint with source test cases. They should give different outputs.

These three patterns are chosen as they suit the study because we are measuring the similarity with a score. We did not select the other four patterns: conjunctive

conditions, disjunctive conditions, disjoint partitions, and complete partitions because we were experimenting with query systems that took keywords or a sentence as inputs to find similar documents. If we use the other four patterns, which can create parts are similar and parts are totally different, the similarity scores might not be able to measure the performance.

5.5 Experimental with Metamorphic Relations pattern of Metamorphic Testing model on Validating Machine learning-based plagiarism detection Systems

This section describes the setup of the experiment. It defines the research objective and research questions of our experiment. Then, it describes the subject systems and data used in the experiment. Thereafter, it discusses detailed experimental design, including environment configuration, measurement, experimental procedure, dataset preparation, and parameter setting. At last, we present our experimental results and further discuss, analyse and interpret the results for research questions.

5.5.1 Research Questions

The main objective of this experiment is to attempt to answer the research question:

RQ1: Can generic MRs of MT, derived from metamorphic relation pattern, validate machine learning-based plagiarism detection systems?

5.5.2 Configuration

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “sklearn”, “gensim” and “SentenceTransformer”. Standard parameters and options were applied to SUTs.

5.5.3 Subject Dataset

This study uses a publicly available dataset from Kaggle³, containing around 50000 English articles. We selected this dataset because, firstly, we are experimenting with the partition difference pattern, and we needed the dataset to be in the English language so we could use words with keywords with opposite meanings to test this pattern. In addition, the English language contains abbreviations of words that can be used to examine Input equivalence patterns.

5.5.4 System Under Test (SUT)

This study focuses on one type of plagiarism detection system, which is a query-based document similarity system. The dataset is transformed into embeddings by four different and common-used techniques, which are “word2vec”, “doc2vec”, “SBERT”, and “tf-idf”. Thus, four SUTs were setup for comparison, and they pick up query inputs and return a list of the most similar documents based on the embeddings and their similar scores, which are computed by the cosine similarity calculation method and its formula is as Equation (5.1).

$$similarity(X,Y) = \frac{X \cdot Y}{\|X\|_2 * \|Y\|_2} \quad (5.1)$$

, where $(\cdot)$ denotes the dot-product of two vectors.

tf-idf is a statistical method to assess the importance of a word to a document by the frequency of a word in different categories. The calculation method of tf-idf is the product of term (word) frequency, which is the occurrence rate of a word in a document, and inverse document frequency, which represents the category discrimination ability of that word [140]. As tf-idf does not have cosine similarity calculation function, this study followed [131]’s implementation to have a machine

³ Dataset is downloaded on 12/31/2022 and from <https://www.kaggle.com/datasets/snapcrack/all-the-news>

learning classification algorithm KNN using cosine similarity method and tf-idf as the text processing model. This study named it “**NN-TFIDF**”.

word2vec is a word embedding method that studies the representation of words in a vector space. It calculates cosine similarity between words through word vector values to understand the semantic relationship between words [120].

doc2Vec is an extension of the “word2vec” model carrying the same idea but solving semantic contextual lack by learning the continuously distributed vector representations for phrases, sentences and documents [43].

Sentence-BERT (SBERT) used a Siamese network structure for semantic textual similarity problems to derive semantically meaningful sentence embeddings. Then, using a similarity measure to calculate the semantic textual similarity between two sentence embeddings [118].

5.5.5 Query

Three source queries are created from a dataset article with ID number “435”. We select all the names of people in the selected article as the first source query (src-q1). The second source query (src-q2) was created by selecting all the numbers in the article. The fifth source query (src-q5) is one sentence extract from the selected article. The other two source queries contain the names of months and the names of weekdays. Contents of the source queries are listed in table 5.1. This study proposed three MRPs on source queries to create nine follow-up queries as follows;

MR1: Shuffling. These queries are created by changing the order of source contents and are expected to return the same outputs. Follow-up query 1.1 (fw1-1) reverses the orders of first names and surnames of src-q1, such as “Cate Blanchett” to “Blanchett Cate”. Follow-up query 1.2 (fw1-2) reverses the order of the whole src-q1. Follow-up query 2.1 (fw2-1) reverses the order of paired numbers in src-q2, such as “384 8” to “8 384”. Follow-up query 2.2 (fw2-2) reverses the order of src-q2. Contents of the follow-up queries are listed in table 5.2.

MR2: Input equivalence. These queries are created by using abbreviations of source contents as they have identical meanings and are expected to return the same outputs. Follow-up query 3.1 (fw3-1) is the abbreviation of src-q3, such as “January” to “Jan”. Follow-up query 4.1 (fw4-1) is the abbreviation of src-q4, such as “Monday” to “Mon”. Contents of the follow-up queries are listed in table 5.3.

MR3: Partition difference. These queries are created using opposite meanings of source contents and are expected to return different results from source queries. Follow-up query 5.1 (fw5-1) uses different nationality, different gender and different number of src-q5. Follow-up query 5.2 (fw5-2) uses only different gender and different number of src-q5. Follow-up query 5.3 (fw5-3) changes singular term into plural term and uses different number of src-q5. Contents of the follow-up queries are listed and highlighted in table 5.4.

Table 5.1: Contents of five source queries

Query ID	CONTENT
src-q1	Cate Blanchett Patrick Catullo Marcia Goldberg Scott Rudin Andrew Upton Nick Kroll John Mulaney
src-q2	384 8 17 129 95 15 40 0 4 1 875 29 2 9
src-q3	January February March April May June July August September October November December
src-q4	Monday Tuesday Wednesday Thursday Friday Saturday Sunday
src-q5	Russian widow who is turning 40 and worried about how to hang on to her home

Table 5.2: Contents of follow-up queries 1.1, 1.2, 2.1 and 2.2

Query ID	CONTENT
fw1-1	Blanchett Cate Catullo Patrick Goldberg Marcia Rudin Scott Upton Andrew Kroll Nick Mulaney John
fw1-2	Mulaney John Kroll Nick Upton Andrew Rudin Scott Goldberg Marcia Catullo Patrick Blanchett Cate
fw2-1	8 384 129 17 15 95 0 40 1 4 29 875 9 2
fw2-2	9 2 29 875 1 4 0 40 15 95 129 17 8 384

Table 5.3: Contents of Follow-up Queries 3.1 and 4.1

Query ID	CONTENT
fw3-1	Jan Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec
fw4-1	Mon Tue Wed Thur Fri Sat Sun

Table 5.4: Contents of Follow-up Queries 5.1, 5.2 and 5.3

Query ID	CONTENT
fw5-1	Chinese widower who is turning 80 and worried about how to hang on to his home
fw5-2	Russian man who is turning 10 and worried about how to hang on to his home
fw5-3	Group who are turning 100 and worried about how to hang on to their home

5.5.6 Implementation

Fig. 5.1 describes the implementation process of this experiment. Firstly, source queries are inputted into four SUTs and the lists of cosine similarity scores of the return documents will be recorded. Secondly, follow-up queries are inputted into four SUTs and the lists of cosine similarity scores of the return documents will be recorded. Thirdly, the study selected the cosine similarity scores of the top 30 documents in the results of source query lists and compared them with the cosine similarity scores of the same 30 documents in the results of follow-up queries lists. Finally, the “Paired T-test” is used for the comparisons. If the differences are not significant for shuffling (MR1) and input equivalence (MR2), then the related MRs can be considered useful properties for validation. If the differences are significant for partition different (MR3), then the related MR can be considered a useful property of validation.

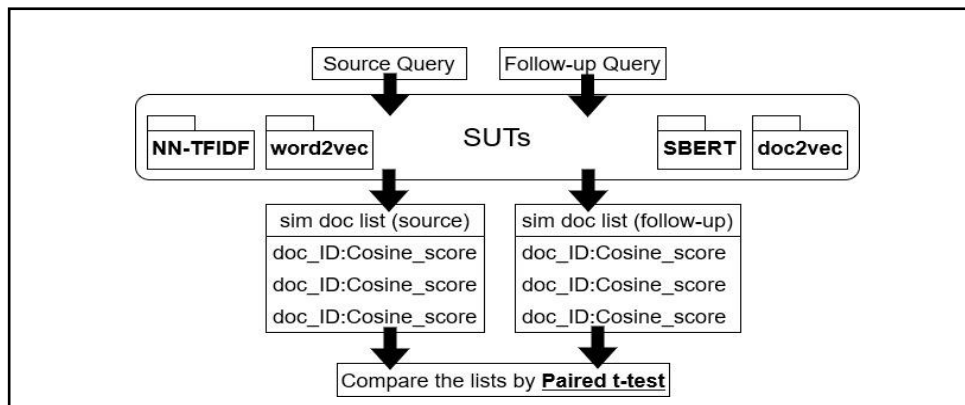


Figure 5.1: The implementation process of the experiment

5.5.7 Experimental Results and Discussion

We studied each of the proposed MRs with all subject systems to evaluate the effectiveness of our proposed approach (RQ1).

5.5.7.1 MR1 - Shuffling

Shuffling with names and numbers works well with NN-TFIDF. All the t-test results of cosine similarity scores between follow-up queries and source queries are “div/0” (Table 5.5) because their means and standard deviations are the same. Both queries returned the same lists of similar documents with the same rankings. NN-TFIDF found the document with ID number 435 (the selected document for source query creation) the most similar document in both source and follow-up queries. As NN-TFIDF only concerns the word frequencies in each document and no semantic relationship is built, the order of the contents in the query should not affect NN-TFIDF. Thus, MR1 is a useful property for validating NN-TFIDF.

Shuffling with names works well with word2vec. Its t-test results of cosine similarity scores between follow-up queries (fw1-1 and fw1-2) and source queries (src-q1) are all the same (Table 5.5). Both queries returned the same lists of similar documents with the same rankings. word2vec also found the document with ID number 435 as the most similar document in src-q1, fw1-1 and fw1-2. Shuffling with numbers works well in “src-q2 vs fw2-1” as the results are identical between src-q2 and fw2-1. However, the t-test result of “src-q2 vs fw2-2” is significantly different (Table 5.5). This might be explained by the fact that word2vec has built a stronger semantic relationship with numbers as the order of the number contents changed more like fw2-2. word2vec model would consider as different search contents. Thus, MR1 with names only is a useful property for validating word2vec.

Shuffling with names works well with doc2vec as both t-test results of “src-q1 vs fw1-1” and “src-q1 vs fw1-2” are not significantly different (Table 5.5), which

means both queries return similar results. However, shuffling with numbers does not work well with doc2vec as t-test results of “src-q2 vs fw2-1” and “src-q2 vs fw2-2” are significantly different (Table 5.5). This might be explained by the fact that doc2vec has built a stronger semantic relationship with numbers as the order of the number contents changed. doc2vec model would consider as different search contents. Thus, MR1 with names only is a useful property for validating doc2vec. Shuffling does not work well with SBERT as all t-test results are significantly different (Table 5.5). As SBERT derived semantically meaningful sentence embeddings, SBERT has built a solid semantic relationship with words and numbers. Therefore, the order of the query contents changed, SBERT would consider as different search contents, and MR1 is too sensitive for validating SBERT.

Table 5.5: MR1 – Paired T-Test results

Paired T-test group	Models	P(T<=t) 2-tail (alpha=0.05)	Reject H0: mean diff. = 0 (p-value < 0.05)
src-q1 vs fw1-1	NN-TFIDF	div/0 ⁴	No
	word2vec	div/0 ⁴	No
	doc2vec	0.0763	No
	SBERT	1.10E-03	Yes
src-q1 vs fw1-2	NN-TFIDF	div/0 ⁴	No
	word2vec	div/0 ⁴	No
	doc2vec	0.7820	No
	SBERT	9.82E-07	Yes
src-q2 vs fw2-1	NN-TFIDF	div/0 ⁴	No
	word2vec	div/0 ⁴	No
	doc2vec	3.32E-04	Yes
	SBERT	2.81E-03	Yes
src-q2 vs fw2-2	NN-TFIDF	div/0 ⁴	No
	word2vec	0.0489	Yes
	doc2vec	9.2E-18	Yes
	SBERT	2.81E-03	Yes

⁴ “div/0” means both lists have all the same values. Their means and standard deviations are also the same.

5.5.7.2 MR2 – Input equivalence

Input equivalence property does not work well with all models except the SBERT model. The t-test result of “src-q3 vs fw3-1” is not significantly different in SBERT, which means both queries return similar results (Table 5.6). As SBERT has built a solid semantic relationship with words, SBERT understands the relationship between the names of months and their abbreviations. Thus, input equivalence of names of months could be used as MR for validating SBERT.

Table 5.6: MR2 – Paired T-Test Results

Paired T-test group	Models	P(T<=t) 2-tail (alpha=0.05)	Reject H0: mean diff. = 0 (p-value < 0.05)
src-q3 vs fw3-1	NN-TFIDF	5.39E-16	Yes
	word2vec	2.53E-24	Yes
	doc2vec	2.39E-03	Yes
	SBERT	0.1940	No
src-q4 vs fw4-1	NN-TFIDF	1.9E-23	Yes
	word2vec	1.22E-34	Yes
	doc2vec	1.92E-09	Yes
	SBERT	3.18E-08	Yes

Table 5.7: MR3 – Paired T-Test Results

Paired T-test group	Models	P(T<=t) 2-tail (alpha=0.05)	Reject H0: mean diff. = 0 (p-value < 0.05)
src-q5 vs fw5-1	NN-TFIDF	6.24E-17	Yes
	word2vec	1.86E-10	Yes
	doc2vec	0.0935	No
	SBERT	2.94E-11	Yes
src-q5 vs fw5-2	NN-TFIDF	2.7E-19	Yes
	word2vec	0.0630	No
	doc2vec	9.59E-05	Yes
	SBERT	2.94E-04	Yes
src-q5 vs fw5-3	NN-TFIDF	6.17E-10	Yes
	word2vec	0.0024	Yes
	doc2vec	0.0680	No
	SBERT	3.04E-15	Yes

5.5.7.3 MR3 – Partition difference

Partition difference property works well with NN-TFIDF and SBERT. T-test results between the source query and all follow-up queries of NN-TFIDF and SBERT are all significantly different, which means follow-up queries return significantly different results (Table 5.7). However, only one out of three t-test results of doc2vec is significantly different, and two out of three t-test results of word2vec are significantly different. As SBERT builds a strong semantic relationship with words, SBERT understands words with opposite meaning and consider them as different search contents. For NN-TFIDF, the terms change in the follow-up queries might also appear less in the documents that the source query considers similar. Thus, MR3 is useful for validating NN-TFIDF and SBERT.

5.5.8 Threats to validity

5.5.8.1 Internal validity

Different input parameters would lead to totally different results. In this experiment, Standard parameters and options (Section 5.5.2) were applied to all subject technologies and systems. We build subject systems using Python libraries (Section 5.5.2) to create the "follow-up" datasets. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Finally, this study focused on the relationship between MR and similarity of the query results mentioned in section 5.5.4. The outcomes were listed and checked to ensure that the measurements mentioned in Section 5.5.4 were calculated accurately. Thus, the potential threat to the internal validity of this study is low.

5.5.8.2 External validity

MT has been widely utilised, and this study is grounded in a generic MT approach. We applied this approach to systems that were featured in other published studies. While the datasets used for evaluation may differ from one case to another, the systems assessed and evaluated by our approach are quite general. Therefore, the potential threat to the external validity of this study is minimal.

5.5.9 Limitations

This experiment has encountered several limitations. Firstly, we only employed three out of seven MRPs from [31]’s list. Secondly, this study only employed one subject dataset. Thirdly, this study only selected datasets in the English language as it suits the proposed MRs.

5.6 Chapter Summary

In this chapter, we conduct an experiment that derives generic MT from MRPs for validating ML-based PDSs. Plagiarism is a serious issue in the academic world, and it is crucial to assess PDS’ performance. This experiment has demonstrated that the proposed approach can systematically and effectively validate different PDSs. We also concluded that models that build a stronger semantic relationship with words or sentences work better with MRPs related to the meaning of contents. Future studies could apply our proposed MRPs or explore other MRPs that are derived to be semantically meaningful for validating different PDSs using models of word, phrase, or sentence embeddings.

The result proved that MRP-based generic MT can serve as a validation tool for ML-based systems, functioning effectively without the need for input from experts or users, which fulfil the research objective to demonstrate the feasibility of applying generic MRs of MT, which are derived from MRP, can validate ML-based systems.

While we recognize the potential of MRPs, it is important to note that MRP “shuffling” may pose challenges when validating pre-trained large language models (LLMs). Shuffling the words in a prompt can lead to unpredictable results from LLMs. Therefore, the next chapter will focus on experimenting with a single pattern: the symmetry MRP referenced by [24]’s study. This pattern aligns with the concept of MRP “input equivalence” discussed in this chapter, emphasizing semantic preservation. We aim to develop a more straightforward and general generic MT approach that can validate various ML-based systems, particularly LLMs.

Chapter 6. Symmetric Metaphoric Relations of Metamorphic Testing for Validating Machine learning-based Systems

In this chapter, we focus on using the symmetry Metamorphic Relation Pattern (MRP) to develop a more straightforward and general generic Metamorphic Testing (MT) approach, which we refer to as symmetric Metaphoric Relations of MT. This approach aims to address the issue of limited participation from experts and users, which can hinder the creation of effective and suitable Metamorphic Relations (MRs) and ultimately affect the validation capacity of MT. We conducted three experiments with different machine learning-based (ML-based) systems to tackle the lack of research interest in validating the performance of ML-based educational systems. All the results showed that generic symmetric MRs of MT derived from symmetry MRP could serve as a validation tool, enhance the MT effectiveness, support simulation MT, and enhance the machine common sense of pre-trained Large Language Models (LLMs).

The rest of this chapter is structured as follows. Section 6.1 introduces the whole work of this chapter. Section 6.2 describes the background of this research. Section

6.3 introduces the motivation. Section 6.4 presents the proposed approach. Section 6.5 provides an overview of three different experiments with symmetry MRP of MT for validating ML-based systems. We conclude this chapter in section 6.6.

6.1 Introduction

Machine learning (ML) models have been applied in software defect prediction, code generation, and maintaining the content quality of question-answering (Q&A) websites. However, evaluating the effectiveness of these models poses a challenge as they tend to provide local optima rather than global optima. In Chapter 5, we demonstrated the feasibility of applying generic MRs of MT, which are derived from metamorphic relation patterns (MRPs), can validate ML-based systems. However, some MRPs, such as MRP “shuffling”, may pose challenges when validating pre-trained large language models (LLMs) because shuffling the words in a prompt can lead to unpredictable results from LLMs. Consequently, this study proposes using symmetric MRs of MT, which is based on the symmetry MRP only, to create a simple and universal generic validation tool for various LLMs.

Firstly, we conducted an empirical experiment involving nine LLMs and two datasets; we assessed the utility of seven symmetric MRs in generating “Follow-up” testing datasets from two distinct “Source” datasets. Our target was to identify the occurrence of violations between the predictions of “Source” and “Follow-up” testing datasets. We then assess the effectiveness of MRs by comparing the non-violated predictions and the ground truth to identify the occurrence of correct and incorrect non-violated predictions. This study has demonstrated that “Duplication of prompts” MRs are the most effective validation tool, cautioning against fixating solely on MR violations to gauge LLM performance. A notable limitation in the effectiveness of MRs was identified as LLMs’ low semantic understanding of prompts. Future studies can focus on exploring different MRs with symmetry, extending the experiment to class-level or line-level code generation, applying

symmetry to prompt engineering, and exploring approaches to turn existing projects into training data systematically.

Secondly, we see the potential of symmetric MRs to further their application in support of simulation metamorphic testing. We conducted another empirical experiment using a publicly available dataset comprising 60000 data points. We defined three groups of Metamorphic Relations (MRGs), consisting of thirteen semantic-preserving MRs, which were then employed to generate “Follow-up” testing datasets based on the original “Source” testing datasets. The pre-trained language models (PLMs) were trained using a separate training dataset. A comparison was made between the predictions of the seven trained PLMs for “Source” and “Follow-up” testing datasets in order to identify instances of violations, which corresponded to inconsistent predictions between the two datasets. If no violation was found, it indicated that the PLM was insensitive to the associate MR; thereby, the MR can be used for validation. In cases where no violation occurred across the entire MRG, non-violation regions were identified and supported simulation metamorphic testing. The results of this experiment demonstrated that the symmetric MRs could effectively serve as a validation tool for content quality classification on Stack Overflow Q&A using PLMs. Four PLMs did not violate the “Uppercase conversion” MRG, and one PLM did not violate the “Duplication” MRG. Furthermore, the absence of violations in the MRGs allowed for the identification of non-violation regions, confirming the ability of the proposed MRs to support simulation metamorphic testing. However, further investigation is required to enhance the semantic comprehension and common sense knowledge of PLMs in order to improve their overall performance.

Lastly, we propose leveraging symmetric MRs to enhance LLMs’ performance by improving their machine common sense. This empirical experiment uses one specific symmetric MR, three LLMs, and a publicly available dataset of labelled Stack Overflow data. We employ the symmetric MR to generate training data that

augments the machine common sense of LLMs. Additionally, we prepare a separate set of training data consisting of labelled Stack Overflow data for comparison purposes. By comparing the results of a common ability test and the predictions made by LLMs trained with different training datasets, we can assess the potential practicality of our proposed approach. Our experimental results demonstrate that a Bert-based LLM trained with MR-generated data outperforms a Bert-based LLM trained solely with regular labelled data. This outcome highlights the effectiveness of symmetric MRs in enhancing LLMs' performance by improving their machine common sense. Subsequent studies can extend our approach to other domains related to education technology and explore additional MRs to further enhance machine common sense.

The results of all three experiments showed that symmetry MRP can be applied more than a validation tool of ML-based systems. It can also be applied to identify the cause of instability of LLMs' outputs, support simulation MT, and work as NLPDA for LLMs to enhance machine common sense.

6.2 Background

In previous chapters, we found that the generic MT model in Chapter 3 can only take on numeric features, and uncertainty-based generic MT in Chapter 4 might require users' input to identify associated uncertainty; thus, we changed our focus to using metamorphic relation pattern (MRP) to derive generic MR of MT. In Chapter 5, we successfully derived generic MRs from the three simplest MRPs from [31]'s list, and this approach is proven to be a robust validation approach without the need for user input. In this chapter, we are inspired by [24]'s study and derive different generic MRs revolving around one pattern, the symmetry MRP, to create a more straightforward and more applicable generic MT approach for validating different ML-based systems, especially pre-trained language models, which are all based on words and sentences.

6.3 Motivation

We see the potential of the symmetry MRP to create a more straightforward and more applicable generic MT approach for validating different ML-based systems, especially LLMs, that suffer from the test oracle problem and lack of research interest in validating the LLMs themselves instead of solely focusing on their performance. Thus, this study aims to demonstrate symmetric MRs of MT as an effective validation tool for ML-based systems through three comprehensive scaled experiments.

6.4 Proposed Approach

6.4.1 Symmetric Metaphoric Relations of Metamorphic Testing

We adopted the symmetric MRs of the MT approach, which is semantic-preserving and inspired by [24]’s study, to validate ML-based systems. [24]’s study introduces the innovative uses of MRs beyond their traditional role in MT. They proposed the symmetry MRP and the change direction MR input pattern. These patterns help derive multiple concrete MRs, which can reveal previously unknown failures in popular applications.

This chapter focuses on the symmetry MRP and the MRs, which operate under the assumption that predictions should remain unaffected when prompts’ semantic meanings are preserved to validate the outputs of LLMs. In other words, positing that the “Source” and “Follow-up” predictions should remain consistent, as the semantic content between “Source” and “Follow-up” does not change. Thus, systems should give the same predictions between the “Source” and the “Follow-up” created by symmetry MRP. Otherwise, there is a violation with MRs, and a fault is revealed.

The main objective is to assess Symmetric MRs derived from symmetry MRP as generic MR of MT for validating ML-based systems. Three experiments are conducted in the following section.

6.5 Experimental Design

This chapter conducted three experiments with symmetric MRs, which are all semantic-preserving. The results demonstrated the capacity of symmetric MRs to validate various types of ML-based systems, which confirmed the potential of symmetric MRs to become an essential validation tool for ML-based systems. In addition, the results showed that symmetric MRs could apply in other areas.

6.5.1 Experiment with Symmetric Metaphoric Relations on validating the code generation LLM

We conduct the first experiment with symmetric MRs on validating the code generation LLM. Code generation is a popular research topic in software engineering [13]. This task generates code by the given natural language description with advanced technologies and techniques, in order to advance the software development process [15, 64]. LLM technology has been utilised for code generation [8, 74, 141], and it is essential to evaluate their performance. However, evaluating code generation LLMs poses a test oracle problem since it is difficult to distinguish correct code generation behaviour from potentially incorrect behaviour. MT was selected for this challenge as it is a widely used technique to alleviate the test oracle problem [19-24]. MT addresses the test oracle problem by examining the expected relationships between input-output pairs in consecutive executions of the systems [142]. These expected relationships are expressed as metamorphic relations (MRs), and if the output results in various software executions violate an MR, a fault is

revealed [24, 31]. We can avoid the test oracle problem and evaluate the models through this approach.

In this experiment, we adopted the effective generic MT approach to validate the stability of code generation LLMs’ outputs. We derived three symmetric MRs from the symmetry MRP inspired by [24]’s study, and these MRs assume that code generations should not be affected as the semantic meanings of the prompts are preserved. We experimented with ten different LLMs on two distinct datasets to simulate software houses using LLM to generate source codes for seen and unseen projects. One dataset contains training and testing data to simulate a seen project, whereas the other dataset only provides testing data to simulate an unseen project.

We used the proposed MRs on the “Source” testing datasets to produce fourteen “Follow-up” testing datasets. We trained eight LLMs with seen project training datasets to create private code generation LLMs, which are referred to as private LLMs. Then, we used eight private LLMs and two public code generation LLMs, which are referred to as public LLMs, to generate source codes based on the prompts from the “Source” testing datasets, resulting in “Source” outputs for seen and unseen projects. We repeated this process with the “Follow-up” testing datasets to generate “Follow-up” outputs for seen and unseen projects. We then compared the two sets of “Source” and “Follow-up” outputs and analysed their differences. A failure is revealed if the relation is violated, which means differences occur between the outputs. Furthermore, by examining the true violations and accuracies of outputs, we could determine the effectiveness of MRs. The results indicate that the symmetric MRs can effectively validate the stability of code generation LLMs’ outputs.

6.5.1.1 Background

This section briefly introduces code generation using large language models, evaluation, and the preliminary knowledge of MT and related works.

6.5.1.1.1 Code Generation

Code generation, also known as code suggestion or text-to-code, is a widely used feature that provides suggested source code based on known source code contexts, traditionally [69, 74]. This task generates code using the given natural language description with advanced technologies and techniques [15]. Code generation is a valuable topic in the software development research domain [13]. Many studies related to code generation have been conducted aiming to increase the productivity of software development and maintain the quality of the source codes [64]. Code generation methodologies include statistical approach, neural symbolic approach [143-147], and pre-trained large language model approach [8, 74, 141]. The employment of large language models, in particular, has demonstrated tremendous promise in completing code and synthesizing code from natural language descriptions [8, 9].

6.5.1.1.2 Large language model (LLM) for code generation and Metamorphic Testing

With the advancements in artificial intelligence, researchers and software developers have shifted their focus towards leveraging artificial intelligence in software engineering tasks, and advanced code generation have been developed because of applying language models [12, 64]. The core of this innovation lies in pre-trained large language models (LLMs), which have shown remarkable effectiveness in both natural and programming language tasks [8, 9, 15]. These models are widely used for solving challenging code processing tasks, such as code generation or code summarization, by providing a flexible interface that enables the synthesis of programs from natural language specifications [16, 65].

LLM-based code completion frameworks such as GitHub Copilot, CodeX, CodeTran, CodeGen, CodeLSTM, RoBERTaCode, GPT and SYNCHROMESH are trained using deep learning over vast quantities of unstructured text and open source

code [9, 13, 15, 16, 74]. Furthermore, it is suggested that self-collaboration among LLMs could potentially enable LLMs to efficiently handle complex real-world tasks that are not readily addressed through direct code generation [141].

Although there are many advantages of using LLMs, companies that use third-party generative AI tools or LLMs, such as ChatGPT, may risk exposing corporate secrets [68]. As language models grow, they also raise safety concerns including but not limited to data contamination, toxic or biased generation, personal information leak, and hallucinations [69].

Stable code generation is essential as if LLM generated codes differently every time, this can introduce risks in the coverage of existing testing procedures. Without validating the stability of the model's outputs, the performance of code-generation LLMs can be unpredictable and lead to unstable systems; however, validating the stability of code generation LLMs' outputs poses a test oracle problem since it is difficult to distinguish correct code-generation behaviour from potentially incorrect behaviour. Thus, MT was chosen to alleviate the test oracle problem. Furthermore, it is plausible to improve the capacity of MT by introducing simulation testing concept [37] and by introducing effectiveness measurement of MT, which refers to measuring how well the evaluation results from MT can reflect the real performance or target system [38].

Motivated by the necessity of more advanced evaluation techniques for code generation LLMs that suffer from the test oracle problem and lack of evaluation approaches focusing on validating the LLM itself instead of solely focusing on the performance of code generation, thus, this study aims to demonstrate generic MT as an effective tool for validating the stability of code generation LLMs' outputs and for explaining the instability through this comprehensive scaled experiment.

6.5.1.2 Research Question

The main objective of the experiment is to demonstrate the feasibility of the proposed MRs for validating the stability of code generation LLMs’ outputs and studying the effectiveness of proposed MRs in explaining the instabilities. This experiment is to attempt to answer three research questions.

RQ1: Can MRs validate the stability of code generation Large Language Models’ outputs? MT is usually conducted when the correct results of individual test cases are unavailable, and the validation capacity of the proposed MRs, which refers to their ability to validate, may vary depending on the outputs of different LLMs. Our goal is to assess the validation capacity of proposed MRs by counting the occurrence of related violations as the measurements of stability under the assumption that there is no ground truth available.

RQ2: How effective are MRs in explaining the instability of code generation Large Language Models’ outputs? We further evaluate the effectiveness of proposed MRs in explaining the instability of LLMs’ outputs by identifying the inconsistencies between performance, “Source” predictions, “Follow-up” predictions, and ground truths related to MRs. These numbers can indicate the effectiveness of the proposed MRs, which refers to measuring how well the evaluation results from MRs can reflect the ground truths related to MRs.

RQ3: What are the causes that make MRs less effective in explaining the instability of code generation Large Language Models’ outputs? When ineffectiveness occurs, we analyse the results with the characteristics of MRs and LLMs to seek the causes that lead to the ineffectiveness.

6.5.1.3 Proposed Metamorphic Relations (MRs)

In this experiment, we adopted the MT approach to validate LLMs and derived three symmetric MRs, which are semantic-preserving, inspired by [31]. [31]’s study introduced innovative uses of MRs beyond their traditional role in MT. They

proposed the “symmetry” MRP and the “change direction” MR input pattern. These patterns help derive multiple concrete MRs, which can reveal previously unknown failures in popular applications. This experiment focuses on the “symmetry” MRP, and our MRs are centered on the concept of symmetry and operate under the assumption that predictions should remain unaffected when the semantic meanings of prompts are preserved (Table 6.1), in order to validate the stability of the LLMs’ outputs. In other words, systems should generate similar codes with “Source” and “Follow-up” prompts as long as the semantic meaning of prompts is preserved. If MRs are not upheld, it indicates a violation, and the associated MR can reveal the instability of the LLM’s outputs.

MR1 - Case conversion of prompts. “Source” and “Follow-up” prompts should mean the same to the code generation LLMs, regardless of whether uppercase or lowercase is used. As only the case format of the prompts is altered, there is no change in the semantic meaning of prompts; thus, code generation LLMs should produce identical outputs with the “Source” and “Follow-up” prompts. For example, we assume two prompts, “loop over files in directory” and “LOOP OVER FILES IN DIRECTORY”, are semantic equivalent to the LLMs, and LLMs should generate similar program statements.

MR2 - Duplication of prompts. “Source” and “Follow-up” prompts should mean the same to the code generation LLMs, regardless of the duplication of the content of prompts. As the contents of prompts are duplicated but not modified, there is no change in the semantic meaning of prompts; thus, code generation LLMs should give the same outputs with the “Source” and “Follow-up” prompts. For example, we assume two prompts, “loop over files in directory” and “loop over files in directory loop over files in directory”, are semantic equivalent to the LLMs, and LLMs should generate similar program statements.

Table 6.1: Descriptions of MRs.

MR	Descriptions
MR1 Case transformation of prompts	MR1.1 - Convert 100% prompt texts to uppercase MR1.2 - Convert 50% prompt texts to uppercase
MR2 Duplication of prompts	MR2.1 - Duplicate the prompt texts two times MR2.2 - Duplicate the prompt texts three times MR2.3 - Duplicate the prompt texts four times
MR3 Paraphrasing of prompts	MR3.1 - Paraphrase prompt texts by online paraphrasing tool MR3.2 - Paraphrase prompt texts by “Mistral-Medium” chatbot from the AI chat platform “Poe”

MR3 - Paraphrasing of prompts. “Source” and “Follow-up” prompts should mean the same to the code generation LLMs, regardless of paraphrasing. In other words, LLMs should consider the contents of prompts have similar semantic meanings; thus, code generation LLMs should give the same outputs with the “Source” and “Follow-up” prompts. For example, we assume two prompts, “loop over files in directory” and “loop through files in directory”, are semantic equivalent to the LLMs, and LLMs will generate similar program statements. In addition, we use one online paraphrasing tool⁵ for MR3.1 and the official “Mistral-Medium” chatbot (<https://poe.com/Mistral-Medium>) from the AI chat platform “Poe”⁶ is chosen for MR3.2.

6.5.1.4 Experimental Settings

6.5.1.4.1 Test Scenario

Inspired by [68], it mentioned that the widespread use of AI chatbots and writing tools could leave companies vulnerable to data leaks and lawsuits. AI chatbots could be a potential area that is exploited by AI companies or hackers to access sensitive corporate information, which has been fed into the chatbots, or perform actions against the company in the future [68]. This study hypothesized that software

⁵ Online paraphrasing tool website is <https://paraphrasing-tool.com/>

⁶ The Poe website is <https://poe.com/>

development houses use private code generation LLMs to mitigate the risk of exposing corporate secrets, such as source code leaks, by restricting access to authorised personnel as the test scenario.

Software development houses must safeguard the company’s software assets while using LLMs to generate source codes for software enhancements and maintenance; however, using private LLMs might not have the same level of performance as public LLMs. Thus, eight private code generation LLMs and two public code generation LLMs are chosen in this study. We selected two separate datasets to simulate the seen and unseen project situation. One dataset includes both training and testing data to simulate a seen project. Whereas the other dataset only provides testing data to simulate an unseen project.

6.5.1.4.2 Subject Under Test (SUTs)

Private pre-trained Large Language Model

Eight pre-trained LLMs from the machine learning community “Hugging Face”⁷ are chosen as private code generation LLMs. We simulated private software houses that downloaded those LLMs from “Hugging Face” and set them up as private LLMs, which are trained privately and only accessed by authorised users. They are referred to as private LLMs throughout our study. Due to the limit of computer resources, this study employed the private LLMs with the little number of parameters.

bert2bert is a text-to-text generation LLM with an encoder-to-decoder framework. The encoder is used to understand the input texts, while the decoder is used to pick up inputs from the encoder and generate text outputs to perform different language tasks, such as code generation. It uses an uncased version of the BERT base language representation model as an encoder and decoder [148]. BERT base language representation model operates as a bidirectional pre-trained language model that processes the raw text inputs by jointly conditioning on both left and right context in

⁷ Hugging Face website is <https://huggingface.co/>

all layers [149]. *bert2bert* is pre-trained and fine-tuned on the “CNN” and “Dailymail” summarisation datasets in the English language [148].

bert2gpt2 is another text-to-text generation LLM with an encoder-to-decoder framework that uses an uncased BERT base language representation model as an encoder and GPT2 base language representation model as a decoder to perform different language tasks, such as code generation [150]. GPT2 is an uni-directional and generative pre-training transformer-based language model created and released by the technology company OpenAI, and it uses language modelling on a large corpus with the ability to process long-range dependencies [151, 152]. GPT2 optimizes the layer normalization, expands the vocabulary size to 50,257, increases the context size from 512 to 1024 tokens, and optimizes with a larger batch size of 512 tokens [152]. In addition, GPT2 is pre-trained on “WebText”, created from scraping web pages, and the dataset roughly contains eight million documents [152]. *bert2gpt2* model is further pre-trained and finetuned with the “id_liputan6” dataset in the Indonesian language [150].

marianCG is a code generation transformer model with an encoder-to-decoder framework developed using the Marian neural machine translation (*marianMT*) model as an encoder and decoder to perform different language tasks, such as code generation [153]. *marianMT* is the core model of the Microsoft Translator, which is the basis for natural language to code translation engine, and uses a sinusoidal positional embedding technique to represent the position of each token in the text, as well as no layer normalization embedding [154]. *marianCG* is pre-trained with “Conala” and “DJANGO” datasets [153, 154].

distilbert2mar is a text-to-text generation LLM with an encoder-to-decoder framework that uses *distilbert*, which is an uncased distilled version of the BERT base language representation model, as encoder and language representation model *marianMT* as a decoder to perform different language tasks, such as code generation [155]. *distilbert* is a pre-trained language representation transformer model trained

with distillation to create a smaller and faster version of BERT [156]. *distilbert2mar* is pre-trained with “BookCorpus” dataset [154, 155, 157].

distilroberta2mar is a text-to-text generation LLM with an encoder-to-decoder framework that uses *distilroberta*, which is a language representation model distilled version of the RoBERTa-base model, as an encoder and language representation model *marianMT* as a decoder to perform different language tasks, such as code generation [158]. Roberta is a transformer model similar to BERT, and it is enhanced with dynamic masking, sentences without next sentence prediction loss, a larger batch size, and a more extensive vocabulary size, which gives it access to more knowledge than other pre-trained language models [159]. *distilroberta* is the distilled version of Roberta, which follows the same training procedure as *distilbert* [160]. *distilroberta2mar* is pre-trained with the “OpenWebTextCorpus” dataset [154, 158, 160].

electra2mar is a text-to-text generation LLM with an encoder-to-decoder framework that uses Electra base language representation model as an encoder and language representation model *marianMT* as a decoder to perform different language tasks, such as code generation [161]. Unlike BERT, Electra uses a more sample-efficient pre-training task called replaced token detection to replace some tokens with plausible alternatives sampled from a small generator network. Then, instead of training a model that predicts the original identities of the corrupted tokens, the model was trained as a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample [162]. *electra2mar* is pre-trained with different datasets, such as “Wikipedia” and “BooksCorpus” [154, 161-163].

luke2mar is a text-to-text generation LLM with an encoder-to-decoder framework that uses LUKE base language representation model as an encoder and language representation model *marianMT* as a decoder to perform different language tasks, such as code generation [164]. LUKE is a pre-trained model designed to create contextualised representations of both words and entities, and is based on the

bidirectional transformer architecture and incorporates an entity-aware self-attention mechanism [165]. This mechanism allows LUKE to treat words and entities as independent tokens, enhancing its ability to understand and process text involving entities [165]. luke2mar is pre-trained with many entity-annotated corpuses obtained from “Wikipedia” [154, 164-166].

pegasus_x is a code generation transformer model that is developed using the PEGASUS-X base language representation model, which is an extension of the PEGASUS model, with additional long input pre-training to handle large size tokens inputs [167, 168]. It is an encoder-to-decoder framework that uses the PEGASUS-X base language representation model as an encoder and decoder to perform different language tasks, such as code generation [167]. pegasus-x is pre-trained and fine-tuned on a variety of datasets, such as “C4”, “arXiv”, “GovReport”, “XSUM”, “CNN” and “DailyMail” [168].

Public pre-trained Large Language Model

Two publicly available pre-trained code generation LLMs are chosen for comparison purposes. They are referred to as public LLMs throughout our study

codeLlama is the first chosen publicly available pre-trained code generation LLM, and it is provided by “poe.com”². “poe.com” is an online platform that provides a wide variety of AI-powered chatbots for use after authentication [135]. codeLlama is an AI-powered chatbot developed by Meta [169] and is based on LLAMA 2 LLM for code generation [170]. LLAMA 2 was trained on a diverse set of publicly available datasets, including “CommonCrawl”, “C4”, “GitHub” repositories, “Wikipedia”, “Books3 and Gutenberg”, “arXiv”, and “Stack Exchange”. This extensive corpus helps LLAMA 2 understand and generate human-like text across a wide range of topics and styles [170-172].

copilot is the second chosen publicly available pre-trained code generation LLM provided by Microsoft. It is based on Microsoft Copilot AI-powered chatbots, and

users can access this chatbot through the sidebar of the Microsoft Edge browser [173, 174]. Based on the user's input prompt and their consent to share data with Microsoft, Microsoft Edge may send relevant data to Microsoft Copilot for response [173, 174]. Microsoft Copilot provides access to powerful AI chatbot that is built on multimodal large language models and text-to-image models, including GPT-4o, which has been pre-trained with a large variety of text sources [174, 175]. It is grounded in the Bing search service to provide responses with the most current information from the web, with verifiable citations for transparency [173, 174, 176]. Thus, copilot combines the power of generative AI with up-to-date information from the vast amount of public web information to perform different language tasks, such as code generation [174].

6.5.1.4.3 Subject Dataset

This study utilizes two distinct publicly available datasets to simulate software houses using LLM to generate source codes for seen and unseen projects: Conala⁸ and MBPP⁹. These datasets contain data pairs of natural language intents and their corresponding Python snippets, which are the prompts and their corresponding source codes, matching this study's intent.

Conala is selected as the seen project dataset, meaning subject LLMs are trained with the Conala dataset to simulate software houses using existing projects' source codes to train the private LLM. The Conala dataset is created by the Carnegie Mellon University NeuLab and Strudel labs for the purpose of testing the generation of code snippets from natural language [177]. This study utilizes the Conala training dataset, which contains 24687 rows of data with intents and snippets, and the Conala validation dataset, which contains 1237 rows of data with intents and snippets, to

⁸ The Conala dataset is downloaded from the Python library "datasets" with the function "load_dataset("data_conala");" on 24/12/2023.

⁹ The MBPP dataset is downloaded from the Python library "datasets" with the function "load_dataset("mbpp", "sanitized");" on 24/12/2023.

train the eight private LLMs. We have selected the first 100 rows of data from Conala testing datasets to create “Source” and “Follow-up” testing datasets for this study.

MBPP is selected as the unseen project dataset, meaning software houses’ LLMs have no prior information about the source codes of these projects. MBPP is a widely used benchmark of Python problems with the prompt for code generation, and the generated code is then tested with the same set of assertions [8]. We select the first 100 rows of data from MBPP testing datasets to create “Source” and “Follow-up” testing datasets for this study.

The Conala and MBPP testing datasets each consist of 100 data pairs, with intents and corresponding Python snippets. These are referred to as the “Source” testing datasets. We applied MRs with “Source” testing datasets to create seven “Follow-up” testing datasets for comparison and analysis.

6.5.1.4.4 Environment

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “transformers”, “pytorch”, “scikit_learn”, for machine learning algorithms. Standard parameters are applied to all LLMs, and the rest of the options are set to default values.

6.5.1.4.5 Measurement

This experiment utilised similarity scores to measure the stability of code generation LLMs’ outputs. It also employed execution success rate and passing rate to measure the performance of code generation LLMs’ outputs.

Similarity

For measuring the similarity between “Source” and “Follow-up” outputs, this study utilizes three scoring metrics: *bleu*, *rouge-l*, and *sacrebleu*. *bleu* is a “corpus-level

metric based on the modified n-gram precision measure” to measure the similarity between sentences [178]. **rouge-l** is a metric from the ROUGE family of metrics and is a “recall-oriented” metric that measures similarity of two sentences by searching for the longest common subsequence between them [178]. **sacrebleu** is python scoring tool which is also using bleu metric and improve the metric by treating bleu “with a bit more reverence” [179]. We employ the average values of these three different metrics to mitigate any bias effects in the scoring process. All three similarity scores range from zero to one hundred. If the average of the three similarity scores is over twenty, the two sentences will be considered similar. We have chosen a cut-off score of twenty because it is based on the minimum bleu score of GoogleCloud translation that the translation is considered understandable [180].

Violation of MR

For measuring the inconsistencies between “Source” and “Follow-up” outputs, we employ the **Violation Rate (VR)** inspired by [19]. VR measures the occurrences of inconsistent results between “Source” and “Follow-up” outputs, which are cases with a similarity score below twenty in this study. If there is no inconsistency between “Source” and “Follow-up” outputs, VR is zero, and LLM is insensitive to the associated MR, which means the MR could be used for validation. Let the total number of test trials be T_{total} and the violated cases be $T_{violated}$. VR is calculated as Eq. (6.1).

$$VR = T_{violated} / T_{total} \quad (6.1)$$

Effectiveness of MR

MT is usually conducted without oracles; thus, the truth results of individual test cases are unavailable. Consequently, the test result reported by MT may not be consistent with the ground truth [38]. In this study, “Source” testing datasets contain truth test results, and the “Follow-up” testing datasets, which were created by the

proposed symmetric MRs and supposed their results are not changed, also contain the same truth test result as “Source”. Thus, we can use the truth test results to assess the effectiveness of MRs. For measuring the effectiveness of MR, we employ the following measurements, which are inspired by Jiang et al. [38] (Table 6.2).

Ground Truth (Truth): These are the ground truth or truth test results of testing datasets.

True Satisfaction (TS): MR has no violation, and the “Source” prediction matches the ground truth. This rate reflects the effectiveness of MR. TS is calculated by counting the occurrences of non-violated cases with consistent results between “Source” outputs and ground truth, which are the cases with similarity scores above twenty between “Source” and “Follow-up” outputs, and between “Source” and ground truth outputs in this study. Let the total number of test trials be T_{total} , and the non-violated and truth-matched cases be $T_{non-violated \& \& truth}$. TS is calculated as Eq. (6.2).

$$TS = T_{non-violated \& \& truth} / T_{total} \quad (6.2)$$

False Satisfaction (FS): MR has no violation, and the “Source” prediction does not match the ground truth. This rate indicates the inconsistency between the system’s behaviour and MR. FS is calculated by counting the occurrences of non-violated cases with inconsistent results between “Source” outputs and ground truth, which are cases with similarity scores above twenty between “Source” and “Follow-up” outputs, and with similarity scores below twenty between “Source” and ground truth outputs in this study. Let the total number of test trials be T_{total} and the non-violated and truth-unmatched cases be $T_{non-violated \& \& not-truth}$. FS is calculated as Eq. (6.3).

$$FS = T_{non-violated \& \& not-truth} / T_{total} \quad (6.3)$$

True Violation (TV): MR has a violation, and the “Source” prediction matches the ground truth. This indicates that the violation is consistent with the match of the

“Source” prediction and the ground truth. TV is calculated by counting the occurrences of violated cases with consistent results between “Source” outputs and ground truth, which are the cases with the similarity scores below twenty between “Source” and “Follow-up” outputs, and with the similarity scores above twenty between “Source” and ground truth outputs in this study. Let the total number of test trials be T_{total} , and the violated and truth-matched cases be $T_{violated\&\&truth}$. TV is calculated as Eq. (6.4).

$$TV = T_{violated\&\&truth} / T_{total} \quad (6.4)$$

False Violation (FV): MR has a violation, and the “Source” prediction does not match the ground truth. This indicates that the violation is inconsistent with the mismatch of “Source” prediction and the ground truth. VR is the total of TV and FV.

FV is calculated by counting the occurrences of violated cases with inconsistent results between “Source” outputs and ground truth, which are the cases with similarity scores below twenty between “Source” and “Follow-up” outputs, and with similarity scores below twenty between “Source” and ground truth outputs in this study. Let the total number of test trials be T_{total} , and the violated and truth-unmatched cases be $T_{violated\&\&not-truth}$. FV is calculated as Eq. (6.5).

$$FV = T_{violated\&\&not-truth} / T_{total} \quad (6.5)$$

Genuine Satisfaction Ratio (GSR): This ratio indicates the proportion of non-violated cases with consistent results between “Source” outputs and ground truth over total violations and non-violated cases with inconsistent results between “Source” outputs and ground truth.

GSR is calculated by dividing TS by the sum of FS, TV, and FV. If the ratio is more than 100%, this indicates that over 50% of cases are TS. On the other hand, if the ratio is less than 100%, this indicates there are less than 50% cases are TS, which

shows the MR introduces true violations (when TV is high) or there are many inconsistencies between “Source” outputs and ground truth outputs (when FV and FS are high). GSR is calculated as Eq. (6.6).

$$GSR = TS / (FS + TV + FV) \quad (6.6)$$

Table 6.2: Illustration of effective measurements.

	Source = Follow-up	Source != Follow-up (VR)
Source = Truth	TS	TV
Source != Truth	FS	FV

Table 6.2 explains that VR is the sum of FV and TV, and the sum of TS and FS is the total of consistent cases between “Source” and “Follow-up”. Table 6.3 illustrates a six cases example of how to calculate VR and GSR. As there are two violated cases out of a total of six cases under the “Violated” column in Table 6.3, VR is 33.33%. On the other hand, two TS cases and four cases belonged to FV, FS and TV under the “Effective measure” column in Table 6.3. Thus, GSR is 50%.

Table 6.3: Illustration of the similarity and effective calculations.

Case	Ground Truth	Source prediction	Follow-up prediction	Violated	Effective measure
1	true	true	true	N	<u>TS</u>
2	false	false	false	N	<u>TS</u>
3	true	false	true	<u>Y</u>	FV
4	false	true	true	N	FS
5	true	false	false	N	FS
6	false	false	true	<u>Y</u>	<u>TV</u>
$VR = T_{violated} / T_{total} = 2/6 = 33.33\%$					
$GSR = TS / (FS + TV + FV) = 2/4 = 50\%$					

Although the outputs between “Source” and “Follow-up” are stable, GSR is lower than 100%, which indicates that non-violated cases with consistent results between “Source” outputs and ground truth are less than 50% of the total of FS, TV and FV. The related MR is low in GSR due to a high number of FS and FV, which means

there are many inconsistencies between “Source” outputs and ground truth outputs, or a high number of TV, which means MR introduces more inconsistencies (Table 6.3). Therefore, effective MR explains the instabilities of LLMs by introducing TV.

Performance of code generation

We also occupied two performance measurements of LLMs to examine how the MRs affect the performance of LLMs.

pass@1: We utilize the pass@K metric [17, 18] to assess the performance of our subject LLMs. We only run one sample-generated code from the “Source” and “Follow-up” prompts, which is a pass@1 test [17].

Additionally, we use the publicly available test cases of MBPP from the CodeT project¹⁰ [181] for calculating this measurement. We executed the codes generated by subject LLMs with “Source” and “Follow-up” prompts of the selected 100 MBPP datasets and tested them with the associated MBPP test cases from CodeT. Then, we counted the number of passed test cases out of 100 test cases as the pass@1 rate of LLMs. This measurement indicates the correctness of LLMs’ generated codes to reflect their performance. Let the total number of test cases be T_{total} , and the passed test cases be T_{passed} . pass@1 is calculated as Eq. (7).

$$pass@1 = T_{passed} / T_{total} \quad (6.7)$$

Execution Success Rate (ESR): As the CodeT project does not provide test cases of Conala, we measure the performance of LLMs by simply executing the codes generated by subject LLMs with “Source” and “Follow-up” prompts of the selected 100 Conala datasets. Then, we count the number of successful executions out of 100 cases as the ESR of LLMs. This measurement indicates the ability of LLMs to generate executable codes. As there is no test case to check against the results of

¹⁰ The MBPP test cases are downloaded on 03/09/2024 from https://github.com/microsoft/CodeT/blob/main/CodeT/data/dataset/mbpp_sanitized_for_test_case_generation.jsonl.

generated codes, ESR will be used with similarity scores to reflect the performance of LLMs with Conala datasets. Let the total number of test trails be T_{total} and the successfully executed cases be $T_{executable}$. ESR is calculated as Eq. (6.8).

$$ESR = T_{executable} / T_{total} \quad (6.8)$$

Suppose the codes generated by subject LLMs with “Follow-up” prompts give lower ESR or pass@1 than those generated by “Source” prompts. In that case, this can be interpreted as the associated MR with more inconsistencies, which is considered an effective MR to explain the instabilities of LLMs’ outputs.

6.5.1.4.6 Implementation

Our experiment includes five steps as follows:

Step 1. We applied the proposed MRs to two “Source” testing datasets and generated fourteen “Follow-up” testing datasets (Fig. 6.1). **Step 2.** Private LLMs train with the training dataset. **Step 3.** Private and public LLMs make predictions on the “Source” testing dataset to create “Source” outputs.

Step 4. Private and public LLMs make predictions on “Follow-up” testing datasets to create “Follow-up” outputs. **Step 5.** We compare two separate sets of “Source” and “Follow-up” outputs and analyse their differences in measurement. A failure is revealed if the relation is violated, which means differences occur between the outputs (Fig. 6.2).

In addition, based on the violations and performance, we can reveal the effectiveness of MRs. Eight trained private LLMs and two public LLMs were tested with two “Source” testing datasets and fourteen “Follow-up” testing datasets. Thus, there were twenty sets of “Source” outputs and one hundred and forty sets of “Follow-up” outputs for comparison in this study.

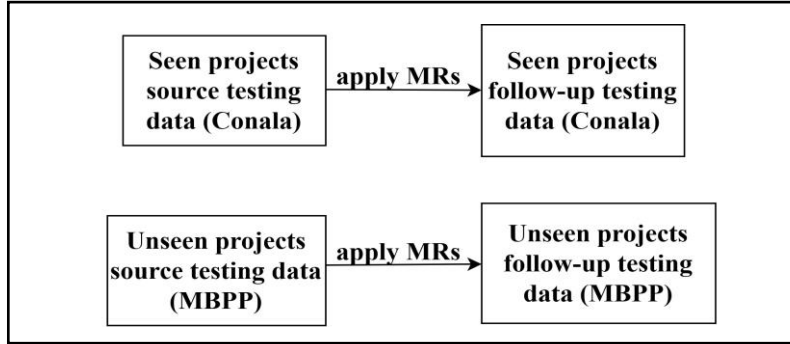


Figure 6.1: Illustration of Step 1.

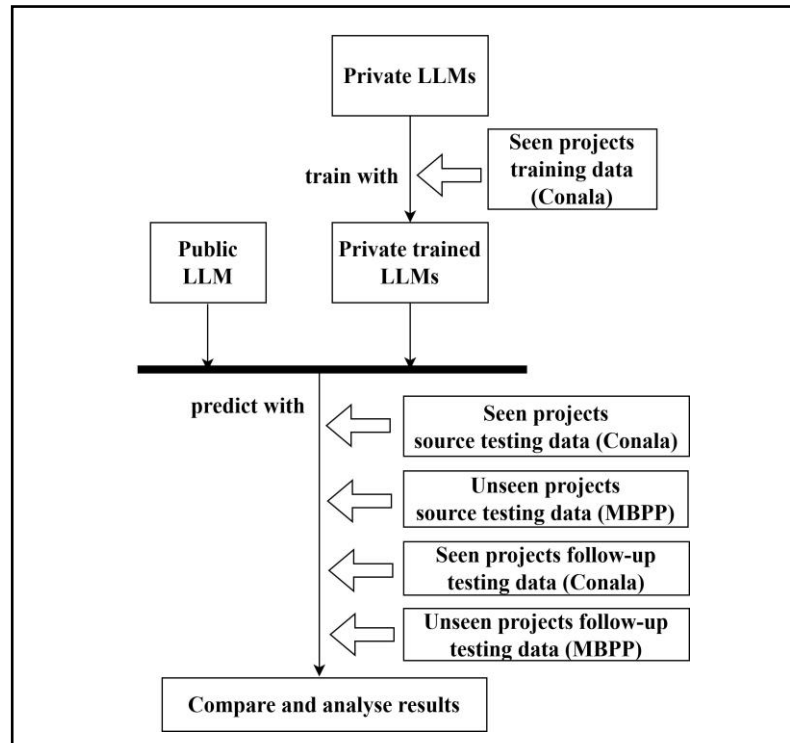


Figure 6.2: Illustration of Step 2 to Step 5.

6.5.1.4.7 Results and discussion

In our study, we presented our experimental results and provided analysis and interpretation for our research questions.

RQ1: Can MRs validate the stability of code generation Large Language models? To address this question, we examined the results of “Zero violation with all MRs” under the simulation that there is no ground truth available.

Zero violation with all MRs. Table 6.4 displays VR values between “Source” and “Follow-up” outputs of the seen project with all subject LLMs and proposed MRs. Table 6.5 displays VR values between the “Source” and “Follow-up” outputs of the unseen project with all subject LLMs and proposed MRs. Tables 6.4 and 6.5 indicate that private LLMs (bert2bert, bert2gpt2, distilbert2mar, and electra2mar) did not violate MR1.1 and MR1.2 in both seen and unseen projects. The results show that the stability of these private LLMs’ outputs is not affected by the case transformation of prompts as the “Source” outputs and the “Follow-up” outputs are similar. bert2bert, bert2gpt2, and distilbert2mar are using uncased version BERT as encoder, so zero violation with case transformation is reasonable for them even though the decoders of bert2gpt2 and distilbert2mar are not uncased version. electra2mar is not an uncased version, but Electra, as the encoder, was trained as a discriminative model that can replace tokens with plausible alternatives. Thus, this technique is able to guard against the influence of case transformation of prompts and keep electra2mar’s outputs stable. Tables 6.4 and 6.5 also show that public LLMs copilot did not violate MR1.1 in the seen project and MR1, MR2 and MR3.2 in the unseen project. The results indicate that case transformation of the prompt will not affect the stability of the copilot’s outputs. Additionally, the copilot performed better in the unseen project, which can be explained by public LLM, which is trained with vast datasets. copilot might have trained with datasets similar to MBPP, so it is reasonable that copilot performed well with the unseen projects. However, codeLlama did not achieve 0% VR with any MR.

Table 6.4: VR values of all LLMs with all MRs in seen project.

LLM	MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	0%	0%	6%	13%	11%	35%	10%
bert2gpt2	0%	0%	12%	18%	15%	46%	21%
distilbert2mar	0%	0%	19%	19%	23%	50%	21%
distilroberta2mar	67%	44%	10%	10%	12%	41%	18%
electra2mar	0%	0%	20%	30%	33%	47%	29%
pegasus_x	68%	37%	11%	16%	20%	45%	21%
luke2mar	64%	39%	12%	24%	25%	47%	21%
marianCG	67%	37%	4%	9%	11%	47%	16%
copilot	0%	1%	1%	1%	2%	1%	3%
codeLlama	17%	5%	11%	1%	4%	13%	7%

Answering RQ1: Four private LLMs did not violate MR1.1 and MR1.2 in both projects, and public LLM copilot did not violate MR1.1 in both projects. These results confirm that the proposed MRs can serve as a validation tool for the stability of code generation LLMs’ outputs under the assumption that there is no ground truth available. In other words, if the outputs of these five LLMs are different between the “Source” and MR1.1, it highlights areas in them that may require further investigation to address the causes of unstable outputs.

Table 6.5: VR values of all LLMs with all MRs in unseen project.

LLM	MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	0%	0%	33%	43%	54%	68%	17%
bert2gpt2	0%	0%	35%	49%	50%	81%	31%
distilbert2mar	0%	0%	34%	40%	41%	61%	24%
distilroberta2mar	90%	69%	34%	41%	44%	59%	29%
electra2mar	0%	0%	36%	58%	65%	52%	16%
pegasus_x	98%	51%	32%	41%	46%	66%	17%
luke2mar	85%	56%	34%	40%	40%	57%	26%
marianCG	88%	66%	24%	31%	31%	64%	22%
copilot	0%	0%	0%	0%	0%	3%	0%
codeLlama	7%	4%	2%	3%	9%	16%	5%

RQ2: How effective are MRs in explaining the instability of code generation Large Language Models’ outputs? To address this question, we examined the outcomes of “Effectiveness and Performance measurements”.

Effectiveness and Performance measurements

The effectiveness and performance measurements have given a different perspective than VR in Tables 6.4 and 6.5. Tables 6.6 to 6.11 display the effectiveness and performance measurements between “Source” and “Follow-up” outputs of unseen and seen projects with all subject LLMs and proposed MRs. Public LLMs have performed better than private LLMs; even some private LLMs achieved zero VR (Tables 6.4 and 6.5), most of them contain significant proportions of FS, TV and FV (Tables 6.6 and 6.8) and the generated codes from private LLMs are not as executable as the codes generated by public LLMs (Tables 6.10 and 6.11).

In unseen project, copilot achieved over 130% GSR and 0% TV and FV with six out of seven MRs (Table 6.6 and 6.7), 84% pass@1 with MR1, MR2.2 and MR2.3, however below 84% pass@1 with MR2.1 and MR3, which are lower than “Source” outputs in the unseen project (Table 6.11). The results indicate that the copilot is insensitive to MR1, MR2.2 and MR2.3 and capable of generating executable codes consistent with ground truth without being affected by MR1, MR2.2 and MR2.3 in the unseen project. On the other hand, the copilot’s unstable outputs, which are the true violations, can be explained by MR2.1 and MR3, which means MR2.1 and MR3 can lead to copilot’s unstable code generations in unseen project (Tables 6.4, 6.5, 6.7 and 6.9). codeLlama achieved over 100% GSR with all MRs except 92.31% with MR3.1 (Table 6.7), and below 61% pass@1 values with all MRs, which are lower than “Source” outputs in the unseen project (Table 6.11). Therefore, all the MRs can contribute to codeLlama’s unstable code generations in the unseen project.

Comparatively, all the private LLMs performed worse than public LLMs regarding effectiveness and performance measurements. All private LLMs have zero pass@1 with all MRs (Table 6.11) in the unseen project because all the generated codes, including the “Source” outputs, are non-executable or unrelated to the prompts. Even bert2bert, bert2gpt2, distilbert2mar, and electra2mar have zero VR

with MR1.1 and MR1.2 in unseen project (Table 6.5), all private LLMs' GSR are below 7%, and pegasus_x, distilroberta2mar, and luke2mar had zero GSR means they have no TS with MR1.1 and MR2 (Table 6.7).

Table 6.6 shows that most private LLMs have high proportions of FS and FV, indicating that "Source" outputs are primarily inconsistent with ground truth in the unseen project (Table 6.12). Despite the low GSR, table 6 shows MR1 did not introduce TV in bert2bert, bert2gpt2, distilbert2mar, and electra2mar, MR2 did not introduce TV in marianCG and luke2mar, and MR3.2 did not introduce TV in distilrobert2mar, electra2mar and marianCG. Although effectiveness results in this study show the proposed MRs did not introduce TV in private LLMs' outputs in the unseen project, the pass@1 values of private LLMs are all zero, which suggests all the proposed MRs can contribute to the private LLMs' unstable outputs in the unseen project.

In seen project, public LLMs achieved over 100% GSR in the seen project with all MRs (Tables 6.8 and 6.9), especially copilot achieved a similar GSR with all MRs as in the unseen project (Tables 6.7 and 6.9). Additionally, copilot achieved over 99% ESR with all MRs except MR1.1 and MR2.2 (Table 6.10). The results show that copilot is capable of generating executable codes that are consistent with ground truth without being affected by all MRs, except MR2.3, which introduced 1% TV in the seen project (Table 6.8) and MR1.1 and MR2.2, which have lower ESR than the "Source" output. Thus, MR1.1, MR2.2, and MR2.3 can contribute to the copilot's unstable outputs in the seen project.

codeLlama achieved better GSR in the seen project, and all GSR values are over 150% with all MRs, except 117.39% with MR1.1 (Tables 6.7 and 6.9). codeLlama also achieved over 94% ESR with all MRs, except MR1.1, MR2.1 and MR2.3 (Table 6.10) in the seen project. However, table 6.8 shows all MRs introduced TV in

codeLlama’s outputs. Therefore, all the proposed MRs can also contribute to the codeLlama’s unstable outputs in the seen project.

Most private LLMs performed better in GSR in seen projects, as bert2gpt2, distilbert2mar, electra_marian, pegasus_x, and luke2mar achieved over 100% GSR with MR1 and MR2 (Table 6.9). However, tables 6.8 and 6.10 show that MR2 and MR3 introduce TV or reduce ESR to lower than “Source” outputs in all private LLMs in the seen project. Tables 6.6 to 6.9 also show that MR1 introduces TV in private LLMs distilroberta2mar, pegasus_x, luke2mar, and marianCG. The results reflect that these four private LLMs are case-sensitive. Thus, all the proposed MRs can lead to private LLMs’ unstable outputs in the seen project.

Answering RQ2: Results indicated that the subject LLMs’ unstable outputs can be explained by “case transformation of prompt”, “duplication of prompt”, and “tokens in prompt changed by paraphrasing”, which introduce true violations to LLMs and affect LLMs to generate executable codes. Therefore, all the proposed MRs can be effective tools for explaining the instability of LLMs’ outputs in both projects.

Table 6.6: TS, FS, TV, and FV values of all LLMs with all MRs in unseen project.

LLM		MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	TS	2%	2%	1%	1%	1%	1%	1%
	FS	98%	98%	66%	56%	45%	31%	82%
	TV	0%	0%	1%	1%	1%	1%	1%
	FV	0%	0%	32%	42%	53%	67%	16%
bert2gpt2	TS	3%	3%	2%	2%	2%	1%	2%
	FS	97%	97%	63%	49%	48%	18%	67%
	TV	0%	0%	1%	1%	1%	2%	1%
	FV	0%	0%	34%	48%	49%	79%	30%
distilbert2mar	TS	4%	4%	3%	3%	3%	3%	3%
	FS	96%	96%	63%	57%	56%	36%	73%
	TV	0%	0%	1%	1%	1%	1%	1%
	FV	0%	0%	33%	39%	40%	60%	23%
distilroberta2mar	TS	0%	1%	0%	0%	0%	1%	2%
	FS	10%	30%	66%	59%	56%	40%	69%
	TV	2%	1%	2%	2%	2%	1%	0%

	FV	88%	68%	32%	39%	42%	58%	29%
electra2mar	TS	6%	6%	4%	4%	4%	2%	6%
	FS	94%	94%	60%	38%	31%	46%	78%
	TV	0%	0%	2%	2%	2%	4%	0%
	FV	0%	0%	34%	56%	63%	48%	16%
pegasus_x	TS	0%	3%	5%	4%	5%	4%	6%
	FS	2%	46%	63%	55%	49%	30%	77%
	TV	6%	3%	1%	2%	1%	2%	0%
	FV	92%	48%	31%	39%	45%	64%	17%
luke2mar	TS	0%	2%	3%	5%	5%	2%	4%
	FS	15%	42%	63%	55%	55%	41%	70%
	TV	5%	3%	2%	0%	0%	3%	1%
	FV	80%	53%	32%	40%	40%	54%	25%
marianCG	TS	1%	2%	4%	4%	3%	2%	4%
	FS	11%	32%	72%	65%	66%	34%	74%
	TV	3%	2%	0%	0%	1%	2%	0%
	FV	85%	64%	24%	31%	30%	62%	22%
copilot	TS	59%	59%	59%	59%	59%	58%	59%
	FS	41%	41%	41%	41%	41%	39%	41%
	TV	0%	0%	0%	0%	0%	1%	0%
	FV	0%	0%	0%	0%	0%	2%	0%
codeLlama	TS	52%	52%	55%	53%	51%	48%	53%
	FS	41%	44%	43%	44%	40%	36%	42%
	TV	3%	3%	0%	2%	4%	7%	2%
	FV	4%	1%	2%	1%	5%	9%	3%

Table 6.7: GSR values of all LLMs with all MRs in unseen project.

LLM	MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	2.04%	2.04%	1.01%	1.01%	1.01%	1.01%	1.01%
bert2gpt2	3.09%	3.09%	2.04%	2.04%	2.04%	1.01%	2.04%
distilbert2mar	4.17%	4.17%	3.09%	3.09%	3.09%	3.09%	3.09%
distilroberta2mar	0.00%	1.01%	0.00%	0.00%	0.00%	1.01%	2.04%
electra2mar	6.38%	6.38%	4.17%	4.17%	4.17%	2.04%	6.38%
pegasus_x	0.00%	3.09%	5.26%	4.17%	5.26%	4.17%	6.38%
luke2mar	0.00%	2.04%	3.09%	5.26%	5.26%	2.04%	4.17%
marianCG	1.01%	2.04%	4.17%	4.17%	3.09%	2.04%	4.17%
copilot	143.90%	143.90%	143.90%	143.90%	143.90%	138.10%	143.90%
codeLlama	108.33%	108.33%	122.22%	112.77%	104.08%	92.31%	112.77%

RQ3: What are the causes that make MR less effective in explaining the instability of code generation Large Language Models' outputs?

To address this research question, we examined with different proposed MRs.

MR1. Tables 6.8 to 6.10 show that MR1 did not introduce TV or reduce ESR to lower than “Source” outputs in bert2gpt2 and distilbert2mar, which means MR1 is ineffective in explaining their instabilities in the seen project. Although they use uncased BERT as an encoder, which gives them an edge in handling case transformation of prompts, bert2bert, which uses uncased BERT as encoder and decoder, has zero ESR with all MRs. So, using uncased BERT as an encoder and mixing with other technologies like GPT2 or marianCG as a decoder could resist the influence of MR1 in the seen project.

Results also show that copilot is insensitive with MR1 in the unseen project (Tables 6, 7 and 6.11). copilot uses the latest GPT-4o generative AI technology and is trained with up-to-date information from the vast amount of public web information and datasets. copilot may have been trained with datasets similar to MBPP (unseen project). Thus, it is reasonable that copilot is insensitive to MR1.

Table 6.8: TS, FS, TV, and FV values of all LLMs with all MRs in seen project.

LLM		MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	TS	45%	45%	44%	43%	41%	33%	43%
	FS	55%	55%	50%	44%	48%	32%	47%
	TV	0%	0%	1%	2%	4%	12%	2%
	FV	0%	0%	5%	11%	7%	23%	8%
bert2gpt ₂	TS	62%	62%	59%	55%	55%	33%	51%
	FS	38%	38%	29%	27%	30%	21%	28%
	TV	0%	0%	3%	7%	7%	29%	11%
	FV	0%	0%	9%	11%	8%	17%	10%
distilbert _{t2mar}	TS	66%	66%	58%	57%	55%	34%	54%
	FS	34%	34%	23%	24%	22%	16%	25%
	TV	0%	0%	8%	9%	11%	32%	12%
	FV	0%	0%	11%	10%	12%	18%	9%
distilrob _{erta2mar}	TS	23%	40%	61%	61%	61%	38%	57%
	FS	10%	16%	29%	29%	27%	21%	25%
	TV	43%	26%	5%	5%	5%	28%	9%
	FV	24%	18%	5%	5%	7%	13%	9%
electr _{a2mar}	TS	59%	59%	52%	46%	42%	36%	46%
	FS	41%	41%	28%	24%	25%	17%	25%
	TV	0%	0%	7%	13%	17%	23%	13%

	FV	0%	0%	13%	17%	16%	24%	16%
pegasus_x	TS	23%	46%	65%	63%	60%	42%	58%
	FS	9%	17%	24%	21%	20%	13%	21%
	TV	45%	22%	3%	5%	8%	26%	10%
	FV	23%	15%	8%	11%	12%	19%	11%
luke2mar	TS	28%	45%	58%	52%	51%	40%	53%
	FS	8%	16%	30%	24%	24%	13%	26%
	TV	35%	18%	5%	11%	12%	23%	10%
	FV	29%	21%	7%	13%	13%	24%	11%
marian CG	TS	23%	47%	72%	70%	70%	42%	66%
	FS	10%	16%	24%	21%	19%	11%	18%
	TV	49%	25%	0%	2%	2%	30%	6%
	FV	18%	12%	4%	7%	9%	17%	10%
copilot	TS	59%	59%	59%	59%	58%	59%	59%
	FS	41%	40%	40%	40%	40%	40%	38%
	TV	0%	0%	0%	0%	1%	0%	0%
	FV	0%	1%	1%	1%	1%	1%	3%
codeLla ma	TS	54%	60%	61%	63%	63%	60%	62%
	FS	29%	35%	28%	36%	33%	27%	31%
	TV	10%	4%	3%	1%	1%	4%	2%
	FV	7%	1%	8%	0%	3%	9%	5%

Table 6.9: GSR values of all LLMs with all MRs in seen project.

LLM	MR1.1	MR1.2	MR2.1	MR2.2	MR2.3	MR3.1	MR3.2
bert2bert	81.82%	81.82%	78.57%	75.44%	69.49%	49.25%	75.44%
bert2gpt2	163.16%	163.16%	143.90%	122.22%	122.22%	49.25%	104.08%
distilbert2mar	194.12%	194.12%	138.10%	132.56%	122.22%	51.52%	117.39%
distilroberta2mar	29.87%	66.67%	156.41%	156.41%	156.41%	61.29%	132.56%
electra2mar	143.90%	143.90%	108.33%	85.19%	72.41%	56.25%	85.19%
pegasus_x	29.87%	85.19%	185.71%	170.27%	150.00%	72.41%	138.10%
luke2mar	38.89%	81.82%	138.10%	108.33%	104.08%	66.67%	112.77%
marianCG	29.87%	88.68%	257.14%	233.33%	233.33%	72.41%	194.12%
copilot	143.90%	143.90%	143.90%	143.90%	138.10%	143.90%	143.90%
codeLlama	117.39%	150.00%	156.41%	170.27%	170.27%	150.00%	163.16%

Table 6.10: ESR values of all LLMs with “Source” and all MRs in seen project

LLM	SRC	MR 1.1	MR 1.2	MR 2.1	MR 2.2	MR 2.3	MR 3.1	MR 3.2
bert2bert	0%	0%	0%	0%	0%	0%	0%	0%
bert2gpt2	1%	1%	1%	0%	0%	0%	0%	1%
distilbert2mar	3%	3%	3%	4%	3%	3%	4%	5%
distilroberta2mar	11%	7%	9%	6%	7%	6%	8%	11%
electra2mar	7%	6%	6%	8%	6%	5%	7%	12%
pegasus_x	6%	7%	11%	3%	6%	3%	14%	7%
luke2mar	4%	3%	6%	4%	5%	3%	10%	3%

marianCG	4%	6%	3%	4%	3%	3%	8%	5%
copilot	99%	81%	99%	99%	98%	99%	99%	99%
codeLlama	94%	84%	96%	93%	95%	88%	97%	94%

Table 6.11: pass@1 values of all LLMs with “Source” and all MRs in unseen project

LLM	SRC	MR 1.1	MR 1.2	MR 2.1	MR 2.2	MR 2.3	MR 3.1	MR 3.2
bert2bert	0%	0%	0%	0%	0%	0%	0%	0%
bert2gpt2	0%	0%	0%	0%	0%	0%	0%	0%
distilbert2mar	0%	0%	0%	0%	0%	0%	0%	0%
distilroberta2mar	0%	0%	0%	0%	0%	0%	0%	0%
electra2mar	0%	0%	0%	0%	0%	0%	0%	0%
pegasus_x	0%	0%	0%	0%	0%	0%	0%	0%
luke2mar	0%	0%	0%	0%	0%	0%	0%	0%
marianCG	0%	0%	0%	0%	0%	0%	0%	0%
copilot	84%	84%	84%	81%	84%	84%	76%	79%
codeLlama	61%	57%	56%	58%	54%	56%	45%	58%

MR2. Tables 6.6, 6.7 and 6.11 show that MR2 did not introduce TV in copilot in the unseen project, and MR2.2 and MR2.3 did not reduce pass@1 to lower than “Source” outputs in copilot in the unseen project. These results show that copilot is insensitive to MR2 in the unseen project as copilot uses the multimodal large language models, including latest GPT-4o generative AI technology, and is trained with up-to-date information from the vast amount of public web information and datasets. It is possible that copilot has been trained with datasets similar to MBPP (unseen project), and the use of multimodal large language models can handle the duplication of prompts. Thus, it is reasonable that copilot is insensitive to MR2.

Table 6.12: The percentage of total cases that have consistent results between “Source” outputs and ground truth

LLM	Source = Truth	
	Seen project	Unseen project
bert2bert	45%	2%
bert2gpt2	62%	3%
distilbert2mar	66%	4%
distilroberta2mar	66%	2%
electra2mar	59%	6%
pegasus_x	68%	6%
luke2mar	63%	5%

marianCG	72%	4%
copilot	59%	59%
codeLlama	64%	55%

MR3. Tables 6.8, 6.9 and 6.10 show that MR3 did not introduce TV or reduce ESR to lower than “Source” outputs in copilot in the seen project only. These results show that copilot can resist the influence of MR3 in the seen project. copilot may use the multimodal large language models to handle prompt paraphrasing. Thus, it is reasonable that the copilot is insensitive to MR3.

Answering RQ3: Results show that using uncased BERT as encoder and mixing with other technologies as decoder can resist the influence of MR1, and copilot can resist the influences of all MRs due to it being pre-trained with a vast number of datasets and using multimodal large language models. Thus, using a mix of LLM technologies and pre-trained with a vast number of datasets can cause the MRs under study to be ineffective. In addition, results imply that all LLMs, except copilot, have low semantic understanding and rely on statistical patterns to infer the prompts [71, 72] because all the prompt transformations are semantic preserving and only copilot can resist MRs’ influences.

Further Analysis

We further analyse with “Effectiveness and Performance measurements of MT”, “MR creation”, “Private and Public code generation LLMs”, “Training data for code generation LLMs” and “Prompt engineering”.

Effectiveness and Performance measurements of MT. The VR results indicate that the validation capacity of MRs is at zero violation, meaning that predictions between “Source” and “Follow-up” are consistent. GSR, TV, TS, FV, and FS results demonstrate the effectiveness of MRs as effectiveness measurements introduce ground truth into the measurements; if TV is not zero, this can reflect MRs introduce

instability of LLMs’ outputs, which can explain the instability by MRs. Especially with the unseen project, effectiveness measurements, GSR, reveal that all private LLMs performed worse than public LLMs in TS, which means private LLMs have many inconsistencies between “Source” outputs and ground truth, despite some private LLMs having zero VR. Thus, effectiveness measurements are essential for understanding the instability of LLMs, which is supported by [38]’s study.

Effectiveness measurements can also provide insights into MRs and LLMs. Despite not achieving zero violation results, private LLM *marianCG* with MR2.1 has the highest GSR and zero TV in the seen project (Tables 6.8 and 6.9). However, performance measurements show a different perspective. In the seen project, public LLMs’ ESRs are better than private LLMs’ ESRs (Table 6.10) because public LLMs generate codes that include “import library” statements, which makes the codes executable by “copy and paste” to the testing environment (which is “Jupyter Notebook” development platform in this study). We further reviewed the reference codes (ground truth) of the seen project and found that most of them are code snippets and are not executable by themselves. Therefore, even though *marianCG* has high GSRs with MR2, the reference truths are not executable, so it has low ESR with MR2 because *marianCG*’s outputs are highly consistent with non-executable “Source” outputs and non-executable ground truths. When the reference code snippets are not executable, it can lead to inaccurate performance measurements. Therefore, it is important to include effectiveness measurements to reflect the instability of the LLMs’ outputs.

MR creation. This study applied [24]’s symmetry MR pattern and created symmetric MRs that can be used to transform prompts and preserve the semantic meanings of prompts. The proposed MRs have been chosen as they are easy to understand and apply to other domains or languages. The results not only show that

the proposed MRs are effective tools for validating the stability of LLMs' outputs, they also show several other insights.

Firstly, results show that all subject LLMs, except copilot, have fewer violations with MR3.2 than MR3.1 (Tables 6.4 to 6.9). This result means LLMs are more sensitive to prompts that were paraphrased by online paraphrasing tools than by AI paraphrasing tools. It can be attributed to online paraphrasing tools are swapping tokens with thesaurus of tokens, such as, "loop over files in directory' source'" to "circle over documents in registry' source'". On the other hand, the AI paraphrasing tool (Mistral) paraphrased the prompt with fewer token changes, such as "loop over files in directory' source'" to "loop through files in directory' source'". Additionally, the stability of six out of seven LLMs' outputs is affected by MR3.1 in both projects (Table 6.6 to 6.11). MR3.1 uses an online paraphrasing tool to swap tokens with a thesaurus, which can impact the stability of highly-trained LLMs. Thus, paraphrasing the prompt by swapping tokens with thesaurus is a valid MR choice because this can explain the instability of six out of seven LLMs' outputs, and the results advised keeping the critical terms used in prompts constant and consistent for the stability of all private LLMs' and codeLlama's outputs.

Secondly, private LLMs bert2bert, bert2gpt2, distilbert2mar, and electra2mar, which used uncased BERT, have zero VR with MR1 as expected (Table 6.4 and 6.5). Although uncased BERT gives them an edge in handling case transformation of prompts, they all have high FS in the unseen project (Tables 6.8 and 6.9). Especially bert2bert, effective measurements show bert2bert cannot achieve over 100% GSR with all MRs in both projects (Tables 6.7 and 6.9) even if it uses uncased BERT in both encoder and decoder. Thus, results justify the selection of MR1; even if the uncased BERT is used, effectiveness and performance measurements can still reveal that bert2bert's "Source" outputs are not executable and inconsistent with ground truth.

Private and Public code generation LLMs. The results indicate that public LLMs perform better in seen and unseen projects, and there are several reasons behind this. Firstly, copilot and codeLlama have been trained with vast number of datasets; therefore, the distinction of unseen and seen datasets in this study might not apply to the public LLMs. However, marianCG has been trained with Conala, and it only performed well in GSR with MR2 in the seen project (Table 6.9). Therefore, public LLMs are more capable of handling the different semantic preserving transformations of prompts by their continuous vast training data and their technologies. As a result, public LLMs have a higher level of semantic understanding than private LLMs.

Secondly, Table 6.10 shows that even though the ground truth of seen projects cannot be executed, public LLMs are powerful enough to add associated “import library” statements and make public LLMs’ generated codes executable. Adding more statements to make the generated codes executable is the reason why public LLMs generated codes that were less similar to the ground truth with the seen project than private LLMs. Private LLMs are trained using the seen dataset, which means they are likely to produce code that resembles the ground truth from the seen dataset. Consequently, if the ground truths are not executable, we can expect that the code generated by private LLMs will also be non-executable. On the other hand, public LLMs’ outputs have included more statements to make their outputs executable despite being less consistent with ground truth than the private LLMs’ outputs. Additionally, table 6.11 shows that only public LLMs can generate codes that can pass the test cases from the CodeT project, while private LLMs can generate consistent outputs with “Source” outputs. However, private LLMs’ generated codes are non-executable as they all have zero pass@1 in the unseen project. The results show that public LLMs can generate codes that are more executable than private LLMs, indicating a higher level of semantic understanding.

Thus, if software houses attempt to employ private code generation LLMs to avoid risk of exposing corporate secrets and enhance software development's productivity at the public code generation LLM level, software houses need to look for solutions to improve the semantic understanding of private code generation LLMs.

Training data for code generation LLMs. The results indicate that all LLMs perform similarly in VR and GSR in the seen project (Tables 6.4 and 6.9). However, public LLMs perform significantly better in the unseen project. The critical distinction between seen and unseen projects is the availability of training data. Without adequate training, it is reasonable that private LLMs perform worse than public LLMs when presented with prompts for the unseen project, as public LLMs have been trained with a continuous vast number of datasets, some of which might be similar to the unseen project. The performance difference shows that providing more different types of training data for code generation could be one of the approaches to enhance the semantic understanding of private LLMs. Although using private LLMs can avoid risk of exposing corporate secrets, vast data training or training targeted at semantic understanding enhancement is needed for private LLMs, as the results in the unseen project show that private LLMs generated incorrect codes.

One potential approach for private LLMs is to use data from existing projects to train them. Cross-project training, which involves using data from other projects, has been successfully employed to enhance the performance of software defect predictions [182-184]. To further enhance the code generation capabilities of private LLMs for future unseen projects, it may be beneficial to consider using diverse code generation related training data from various existing projects from different sources. Additionally, specific projects within one's own company could be leveraged to train private LLMs to be more specialised in a particular method of code generation.

Public LLMs codeLlama and copilot optimized its training by using various types of data from different sources, such as source codes, discussions about source codes,

code snippets, and natural language questions and answers; and various approaches, including code infilling and long context fine-tuning [170-172, 175]. As a result, public LLMs are capable of handling the unseen project. Future studies can apply the approaches of codeLlama [170] to systematically collect and convert different sources into training data, which can enhance the semantic understanding of private LLMs.

Although marianCG was trained with Conala, it only had better results than public LLMs in the seen project with MR2. Thus, public LLMs show a higher level of semantic understanding of prompts, and training that can improve the semantic understanding of LLMs can be helpful. As the results have demonstrated that the proposed symmetric MRs are effective and semantic preserving, they can also create different semantic preserving training datasets based on “Source” training datasets for further training, like data augmentation, to enhance the semantic understanding even in the situation of testing data scarcity. Thus, using multiple sources to create more training datasets for private LLMs is a potential approach to improve the semantic understanding of private code generation LLMs.

Prompt engineering. The generation results of the prompt must be justifiable, explainable, and verifiable [185]. It is important to explore how the prompts are formulated to guide LLMs to replicate the code generation because most LLMs are challenging to update or modify. Therefore, prompt engineering plays a critical role in ensuring stable and reliable code generation.

Tables 6.4 to 6.11 demonstrate that, except for MR3.1, the proposed MRs can serve as effective validation tools for the stability of LLMs’ outputs. These results also suggest that the concept of symmetry can be applied to prompt engineering. Symmetric MRs are expected to maintain stable code generation as long as the semantic meanings of the prompts remain consistent. Therefore, prompts created by

symmetric MRs can generate source codes that are justifiable, explainable, and verifiable.

The results of MR3.1 show that it is essential to identify the key terms of prompt for improving the stability. Thus, paraphrasing with the thesaurus swapping of tokens in prompts and testing with our approach can assist in identifying the critical tokens of prompts that cannot be changed to improve stability.

Lastly, more descriptions and more structures in the prompt might be needed for private LLMs as the pass@1 results show they provide inexecutable codes or codes that are not related to the prompts (Tables 6.10 and 6.11), which reflects that private LLMs do not understand the prompts in the unseen project.

6.5.1.4.8 Threats to Validity

Internal validity. One internal threat to the experiment is related to parameter configuration. Standard parameters were applied to all LLMs, and we built eight private LLMs with the pre-trained models provided by the machine learning community “Hugging Face” and the libraries of “Python”. The public LLMs are unconfigurable and available on “Poe.com” website and “Microsoft Edge” browser. Additionally, we utilised Microsoft Excel to create the “Follow-up” testing datasets. These tools and technologies are reliable and widely used. White-box testing has also been conducted to ensure error-free source codes. Another concern is the measurements of this study. We specifically focused on the relationship between MRs and measurements, and the prediction outcomes of all LLMs were listed and checked to ensure measurements were calculated accurately. At last, the transformations of the prompt might lead to incorrect code generations. All the proposed MRs are inspired by Z. Q. Zhou et al. [24]’s “symmetry” MR pattern, and results show LLMs can generate “Follow-up” executable outputs that are not violated with “Source” outputs and ground truths, which indicates that the choice of the proposed MRs is valid as they are capable of preserving the semantic meaning

of prompts and are easy to understand and apply. We believe the chance of MRs creating invalidity is low.

External validity. One external threat is the generality of the proposed MRs. This study uses symmetric MRs, which are semantic-preserving, generic and applicable to any experiment using the English language. This study has chosen two publicly available datasets with over 20000 data pairs, and nine different LLMs were employed to run trials with seven MRs. The results should be generalised, and the effect of datasets on the effectiveness of the proposed MRs should be small. Lastly, the measurements used in this study are rather general and replicable; thus, the potential threat to the external validity of this study is low.

Construct validity. The independent variables of this study are prediction outcomes of the “Source” and “Follow-up” testing datasets of ten subject code generation LLMs, of which eight were trained by the same training dataset as private LLMs, and two are publicly available LLMs. The dependent variable is the differences between the independent variables, which are the differences between the prediction outcomes of “Source” and “Follow-up” testing datasets of LLMs, which is the VR. In addition, this study uses the accuracy of “Source” predictions and dependent variables to assess the effectiveness of MRs: the GSR, TV, FV, TS and FS. At last, this study uses the prediction outcomes of “Source” and “Follow-up”, which are the generated codes, to assess the performance of MRs: the ESR and pass@1.

This study employed a score of twenty as the cut-off score to classify the “Source” and “Follow-up” outputs as similar. The chosen score was based on the minimum bleu score of GoogleCloud translation, indicating that the translation is understandable [180]. However, it is essential to note that the cut-off score of twenty was a heuristic choice. Despite this, this cut-off score is applied to all LLMs with all the proposed MRs, ensuring that the experimental result contains no bias. This study

aims to validate the stability of code generation of LLMs' outputs. Thus, we believe using this cut-off score is appropriate.

As this study focuses on validating the stability of LLMs' outputs and measuring the effectiveness of MRs, VR could show the violations. In contrast, GSR, TV, FV, TS and FS could show the effectiveness, and ESR and pass@1 could show how the MRs affect the performance of LLMs. Suppose there is no violation between "Source" and "Follow-up" prediction outcomes; it means there is no inconsistent prediction between them. This result indicates that the associated LLM is insensitive to the related MR, and the MR could be used for validation. A violation with the MR could reveal an error in LLM because the related LLM should be insensitive to the MR. If VR is not zero and MR introduces TV, meaning "Source" predictions are consistent with ground truth and inconsistent with "Follow-up" predictions, this is interpreted as the related MR is effective as it can explain the instability of LLMs' output.

Furthermore, the performance measurements of "Follow-up" outputs are lower than the "Source" outputs; this also can be interpreted as the associated MR is effective as it can explain the instability of LLMs' output. Those can correctly measure the intent of this study. To sum up, the construct validity of the independent variables and dependent variables of this article should be acceptable.

6.5.1.4.9 Limitations

Due to difficulties in accessing OpenAI's ChatGPT-4 and Google's Gemini technology in Hong Kong, this study did not consider those latest AI technologies. In addition, we only selected Python programming language-based Conala and MBPP as subject datasets.

6.5.1.5 Conclusion

This experiment demonstrated that the proposed MRs could serve as a validation tool with violation measurements for the stability of code generation LLMs' outputs

under the assumption that there is no ground truth available. One public LLM and four private LLMs did not violate MR1.1 (Case transformation of prompts) in the seen or unseen project, suggesting that if the outputs of these five LLMs violate MR1.1, there are faults in those LLMs systems needed to be investigated that cause the unstable outputs.

The effectiveness and performance results, which include identifying the inconsistencies between performance, “Source” predictions, “Follow-up” predictions, and ground truths related to MRs, indicated that proposed MRs are indeed effective tools for explaining the instability of LLM’s outputs by “Case transformation of prompts”, “Duplication of prompts”, and “Paraphrasing of prompts”. As MRs are semantic preserving, results also show that enhancing the semantic understanding of prompts is required by all private LLMs and codeLlama for better code generation. The main factors limiting the effectiveness of MRs in identifying the causes of instability are LLMs use a mix of LLM technologies and LLMs are pre-trained with the vast number of datasets. On the other hand, the factors imply that LLMs have a low semantic understanding of prompts because all the prompt transformations are semantic preserving, and only the copilot can resist MRs’ influences.

Additionally, results show that, firstly, effectiveness measurements are essential for understanding the instability of LLMs, especially when performance measurements cannot give an accurate reflection. Secondly, the proposed MR choices are valid. Thirdly, it is necessary to look for solutions to improve the semantic understanding of private code generation LLMs, in order to enhance their performance. Using multiple sources, such as existing projects, source codes, discussions about source codes, code snippets, and natural language questions and answers, to create more training datasets for private LLMs is a potential approach. Fourthly, proposed symmetric MRs can be used for prompt engineering and

identifying key tokens of prompts. Lastly, private LLMs require prompts that include more detailed descriptions and structures, in order to better understand the prompts.

6.5.2 Experiment with Symmetric Metaphoric Relations on validating the content quality classification of Pre-trained Language Models

After the first experiment, we confirmed that Symmetric MRs could serve as a validation tool for code generation LLMs. This second experiment with symmetric MRs is aiming to validate the content quality classification of Pre-trained Language Models (PLMs), which is different from the previous experiment. Question-answering (Q&A) websites, which serve as a dynamic self-learning platform, provide a source of learning or solution searching for different users [7]. Stack Overflow and Quora are among the largest and most popular self-learning platforms for different levels of computer and software practitioners [7, 50, 186]. Given their significance, the content quality of the Q&A websites must be maintained to remain popular and provide quality support to their users [7].

Machine learning (ML) is a cutting-edge technology that has been applied to maintain the quality of Q&A websites by applying PLMs to either replying the users' questions or classifying the content quality of the sites automatically [7, 49, 50]. Although PLMs are capable of performing the replying and classification tasks, a robust evaluation approach is required to ensure the correctness of PLMs [53].

Evaluating the performance of ML models, including PLMs, is a test oracle problem since ML models can deliver local optima rather than global optima. Metamorphic Testing (MT) was selected for this challenge as it is a widely used technique to alleviate the test oracle problem [19-24]. MT addresses the test oracle problem by examining the expected relationships between input-output pairs in consecutive executions of the systems [142]. These expected relationships are expressed as metamorphic relations (MRs), and if the output results in various

software executions violating an MR, a fault is revealed [24, 31]. We can avoid the test oracle problem and evaluate the models through this approach.

In this experiment, we adopted the generic MT approach to validate PLMs for content quality classification of Q&A websites. We derived three groups of semantic-preserving MRs inspired by [24], and these MR groups (MRGs) assume that predictions should not be affected as the semantic meanings are preserved. We experimented with four different PLMs on labelled datasets containing 60000 Stack Overflow questions with quality ratings. We applied MRs to generate thirteen follow-up testing datasets for testing. The source and follow-up predictions of PLMs were compared, and if there was no violation, meaning there was no inconsistent prediction between the two datasets, the PLM was insensitive to the related MR and that MR could be used for validation. If the PLM was insensitive to data transformation, then any violation of MR could be interpreted as a fault in the model. Furthermore, if there was no violation with the entire MRG, non-violation regions were identified to support simulation MT. The results show that the proposed MRs can effectively validate PLMs on classifying the content quality of Stack Overflow Q&A and can support simulation MT by identifying the non-violation regions.

6.5.2.1 Background

This section briefly introduces question-answering (Q&A) websites and how they apply machine learning approach.

6.5.2.1.1 Question-answering (Q&A) websites

Question-answering (Q&A) is a crucial technology in the field of human-computer interaction [53], and Q&A websites provide a self-learning platform for many users. Quora and Stack Overflow are among the largest and most popular Q&A self-learning platforms for software-related issues [7, 50, 186]. These platforms contain different user-provided content ranging from simple guidelines to highly

sophisticated and experienced answers, serving as valuable sources of learning and solutions for users at different levels [7]. Thus, their content quality must be maintained to protect the sites' reputation, attract more visitors, provide a better user experience, encourage experts answering more questions, and increase search result rankings of the sites [7]. Maintaining the content quality is accomplished by keeping questions clean and clear, removing unpleasant content and duplicated questions, and safeguarding high-quality content from unwanted changes [7].

6.5.2.1.2 Apply Machine Learning Approach to Q&A

ML models, such as PLMs, have been applied to maintain the content quality of Q&A websites and respond to questions across various Q&A platforms [49, 50, 58-60]. Furthermore, machine learning models have been applied to generate hints for the question, aiding users in getting the final answers and clarifying the questions [62]. In addition, they have been used to construct a multimodal medical Q&A system with medical text and image data [61].

PLMs have been utilised on Medical Q&A sites for answering medical questions; although the performance of PLMs is reasonably acceptable, human reviews and evaluation frameworks are required for creating safe and helpful clinically used PLMs [49]. Generative Pretrained Transformers (GPT), which is a PLM developed by the company OpenAI, has been used to reply to Stack Overflow questions; however, human responses to Stack Overflow questions are still better than PLMs [50].

Despite human responses are better at answering questions, it is challenging to maintain the content quality of Q&A from community support websites, such as Stack Overflow, and using PLMs has demonstrated the ability to automate moderation and content quality prediction [7, 70]. PLMs were also applied to evaluate and examine the content quality of health-related Q&A sites, and the results were acceptable [59, 60]. Additionally, PLMs have been employed to identify

duplicate questions on Game Development Stack Exchange and Stack Overflow Q&A websites to enhance content quality [63].

Overall, PLMs exhibit good performance on Q&A tasks and have the potential for further enhancing their proficiency in social, event, and temporal common sense knowledge in the future; however, there is a need for an evaluation method to ensure the accurate assessment of PLMs’ capabilities and limitations [53].

6.5.2.2 Research Question

Validation of machine learning (ML) models, such as PLMs, is a test oracle problem since machine learning models deliver local optima rather than global optima [19, 22-24]. To address this issue, a systematic and practical approach for validating PLMs is needed, and MT can be employed to mitigate the test oracle problem [19, 22-24]. Furthermore, [37] have incorporated the simulation testing concept into MT and introduced the concept of MR-violation regions to enable testers or researchers to determine whether the test case is violated or not by the location of the test case. Therefore, this experiment aims to demonstrate MT's practicality in validating PLMs and supporting simulation MT through a comprehensive scaled experiment. The main objective of this experiment is to attempt to answer two research questions.

RQ4: Can MRs validate the content quality classification of Stack Overflow problems using Pretrained Language Models?

RQ5: Can MRs support simulation metamorphic testing for Pretrained Language Models?

6.5.2.3 Proposed Metamorphic Relations (MRs)

In this study, we adopted the MT approach to validate PLMs and derived three semantic-preserving MR groups (MRGs) inspired by [24]. These relation groups revolve around the principle of symmetry, positing that the quality predictions for content—specifically, for “Source” and “Follow-up” questions on Stack Overflow—

should remain consistent, as the semantic content does not change (Table 6.13). Thus, systems should classify the content quality of “Source” Stack Overflow questions and the “Follow-up” Stack Overflow questions created by MRGs with the same quality rating. MRGs should be held. Otherwise, there is a violation with MRGs and a fault in the PLM.

6.5.2.3.1 MRG1 – Duplication

“Source” and “Follow-up” prediction results should be consistent regardless of duplicating the content of title texts. Due to the limited computer resources, this experiment only duplicates the content of title texts (not title and body text). As the contents of title texts are duplicated but not modified, there is no change in semantic meaning; thus, they should give the same outputs. For example, we assume two sentences, “I don’t know” and “I don’t know I don’t know”, are semantic equivalent to the models, and models will classify both sentences with the same label. We create five associated MRs duplicating the content of the title texts at different times, and they are MR1.1, MR1.2, MR1.3, MR1.4, and MR1.5 (Table 6.13).

6.5.2.3.2 MRG2 – Abbreviation replacement

“Source” and “Follow-up” prediction results should be consistent regardless of whether terms are presented in their abbreviated form or full form. This consistency is expected as the transformation involves expanding abbreviations in the title and body texts to their full forms without altering the overall semantic meaning. Thus, they should give the same outputs. For example, we assume two sentences, “I don’t know” and “I do not know”, are semantic equivalent to the models, and models will classify both sentences with the same label. We create four associated MRs that replace different numbers of abbreviations into full forms, and they are MR2.1, MR2.2, MR2.3, and MR2.4 (Table 6.13).

Table 6.13: Table of the proposed semantic-preserving MRs.

MR group (MRG)	Descriptions
MRG1: Duplication	MR1.1 – duplicate the content of the title two times MR1.2 – duplicate the content of the title three times MR1.3 – duplicate the content of the title four times MR1.4 – duplicate the content of the title five times MR1.5 – duplicate the content of the title six times
MRG2: Abbreviation replacement ¹¹	MR2.1 – Replace [don't, doesn't, didn't] with [do not, does not, did not] MR2.2 – Replace [don't, doesn't, didn't, can't, couldn't] with [do not, does not, did not, cannot, could not] MR2.3 – Replace [don't, doesn't, didn't, can't, couldn't, won't, wouldn't] with [do not, does not, did not, cannot, could not, will not, would not] MR2.4 – Replace [don't, doesn't, didn't, can't, couldn't, won't, wouldn't, shalln't, shouldn't, isn't, wasn't, aren't, weren't] with [do not, does not, did not, cannot, could not, will not, would not, shall not, should not, is not, was not, are not, were not]
MRG3: Uppercase conversion	MR3.1 – Convert 25% texts to uppercase MR3.2 – Convert 50% texts to uppercase MR3.3 – Convert 75% texts to uppercase MR3.4 – Convert 100% texts to uppercase

6.5.2.3.3 MRG3 – Uppercase conversion

“Source” and “Follow-up” prediction results should be consistent regardless of whether uppercase or lowercase is used. As only the case format of the title and body texts are changed, there is no change in semantic meaning; thus, they should give the same outputs. For example, we assume two sentences, “I don’t know” and “I DON’T KNOW”, are semantic equivalent to the models, and models will classify both sentences with the same label. We create four associated MRs that convert texts from lowercase into uppercase at different percentages of texts in the title and the body texts, and they are MR3.1, MR3.2, MR3.3, and MR3.4 (Table 6.13).

¹¹ This experiment does not check the spelling errors, such as “cant”, “dont”.

6.5.2.4 Experimental Setting

In this section, we evaluated the effectiveness of the proposed semantic-preserving MRs group through this experiment’s design and implementation.

6.5.2.4.1 Test scenario

Inspired by [7], this study used the self-learning Q&A website “Stack Overflow” as the test scenario. It is crucial to maintain the content quality of the Q&A site, and using PLMs to classify content quality is a time-saving approach for maintaining and improving the quality of the site [7]. Four PLMs are chosen in this study as the classifiers of the Stack Overflow questions.

6.5.2.4.2 Subject pretrained language model

This experiment involved four PLMs provided by the machine learning community “Hugging Face”¹².

XLNet was firstly introduced by [187] and is pretrained on a large-scale dataset, including Wikipedia, BookCorpus, Giga5, ClueWeb, and Common Crawl. It is an unsupervised generalised pretraining learning method based on a permutation-based language modelling objective and employs the Transformer-XL model as the core model to preserve the advantages of the autoregressive language model and bidirectional model [7, 71, 187, 188]. XLNet has been applied in language tasks that use sentences to make decisions, such as text generation, text classification and question answering, and this study uses the cased version [7, 188].

XLNet-Roberta. Roberta is a transformer model similar to BERT, which operates as a bidirectional PLM that processes from “unlabeled text by jointly conditioning on both left and right context in all layers” [149]. XLNet-Roberta is enhanced with

¹² Hugging Face website is <https://huggingface.co/>

dynamic masking, sentences without next sentence prediction loss, a larger batch size, and a more extensive vocabulary size, which gives it access to more knowledge than other PLMs. As for XLM-Roberta, it is a multilingual cased version of Roberta and is pretrained with filtered CommonCrawl data which is at the size of 2.5 terabytes and contains over 100 languages [71, 189, 190].

T5, which is developed by Google and pretrained on the Colossal Clean Crawled Corpus, is an encoder-decoder transfer learning model for natural language processing that converts every language problem into a unified text-to-text format [191-193]. The model has demonstrated proficiency across various language tasks, including question-answering and text classification [192]. This study utilises version 1.1 of T5.

GPT1 is an uni-directional language model and the pioneering generative pretraining transformer-based language model created and released by the technology company “OpenAI”, and it uses language modelling on a large corpus with the ability to process long-range dependencies [71, 151, 194]. GPT1 is pretrained with the BooksCorpus dataset, containing over 7,000 unique unpublished books and has proven to solve discriminative tasks such as question answering, semantic similarity assessment, entailment determination, and text classification [151].

6.5.2.4.3 Subject Dataset

This study utilises a publicly available dataset from Kaggle¹³, which was created to study the content quality classification of Stack Overflow questions [7]. The dataset contains 60000 Stack Overflow questions in English and serves the intent of this study. Questions are classified into three labels: “High quality questions that require

¹³ The dataset is downloaded on 23/09/2023 from <https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questions-with-quality-rate>

no moderation action” (HQ), “Low-quality questions that require community edits” (LQ_EDIT), and “Low-quality questions that were closed by the community without a single edit” (LQ_CLOSE) [7, p. 147].

For this study, the training dataset includes 45000 Stack Overflow questions with labels for training PLMs. The testing dataset contains 15000 Stack Overflow questions with labels and is the “Source” testing dataset. We applied MRGs with “Source” testing datasets to create thirteen “Follow-up” testing datasets for comparison. In addition, datasets are cleansed and all set to lowercase before training and testing.

6.5.2.4.4 Environment

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “transformers”, “tensorflow”, “scikit_learn”, for machine learning algorithms. Standard parameters are applied to all PLMs, and the rest of the options are set to default values.

6.5.2.4.5 Measurement

This experiment measures the re-labelling rate (RR), inspired by [19], to measure the differences between the source and follow-up outputs. Source outputs were the prediction outcomes of PLMs on the “Source” testing dataset, whereas follow-up outputs were the prediction outcomes of PLMs on the “Follow-up” testing datasets. Suppose there is no violation, which is prediction inconsistency, between source outputs and follow-up outputs. In that case, RR is zero, and the PLM is insensitive to the associated MR, which means the MR could be used for validation.

Let “Source” testing dataset be $T_s = [x_1, x_2 \dots x_n]$ and “Follow-up” testing dataset be $T_f = [x_{f1}, x_{f2} \dots x_{fn}]$. PLM was trained by the training dataset. The prediction outcomes of PLM with T_s be $Y_s = [y_{s1}, y_{s2} \dots y_{sn}]$, and the prediction outcomes of

PLM with Tf be $Yf = [yf1, yf2 \dots yfn]$. Let dif be the number of cases with label differences between Ys and Yf . RR is calculated as Equation (6.9).

$$RR = dif / \text{total size of testing dataset} \quad (6.9)$$

The above calculation indicates that no difference between Ys and Yf if RR is zero, which is considered a zero violation of this MR. If RR is larger than zero, then violations have occurred. Concerning validation, if RR is zero, no violation is caused by MR, and vice versa.

As the subject datasets are labelled, we also employ the prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) [73]. These measurements can show any relationship between MRs and prediction accuracy. Let Tp be true positives that are the actual positives and correctly predicted, Tn be true negatives that are the actual negatives and correctly predicted, Fp be false positives that are the actual negatives and wrongly predicted as positives, and Fn be false negatives are the actual positives and wrongly predicted negatives. ACC is calculated as Equation (6.10), and MCC is calculated as Equation (6.11)

$$ACC = (Tp + Tn) / (Tp + Tn + Fp + Fn) \quad (6.10)$$

$$MCC = \frac{(Tp \times Tn - Fp \times Fn)}{\sqrt{((Tp + Fp) \times (Tp + Fn)) \times ((Tn + Fp) \times (Tn + Fn))}} \quad (6.11)$$

6.5.2.4.6 Implementation

Our experiment includes six steps as follows:

Step 1. We apply the proposed MRs to the “Source” testing dataset and generated thirteen “Follow-up” testing datasets.

Step 2. We train the subject PLMs with a training dataset containing 45000 Stack Overflow questions with labels.

Step 3. PLMs make predictions on the “Source” testing dataset containing 15000 Stack Overflow questions with labels to create source outputs.

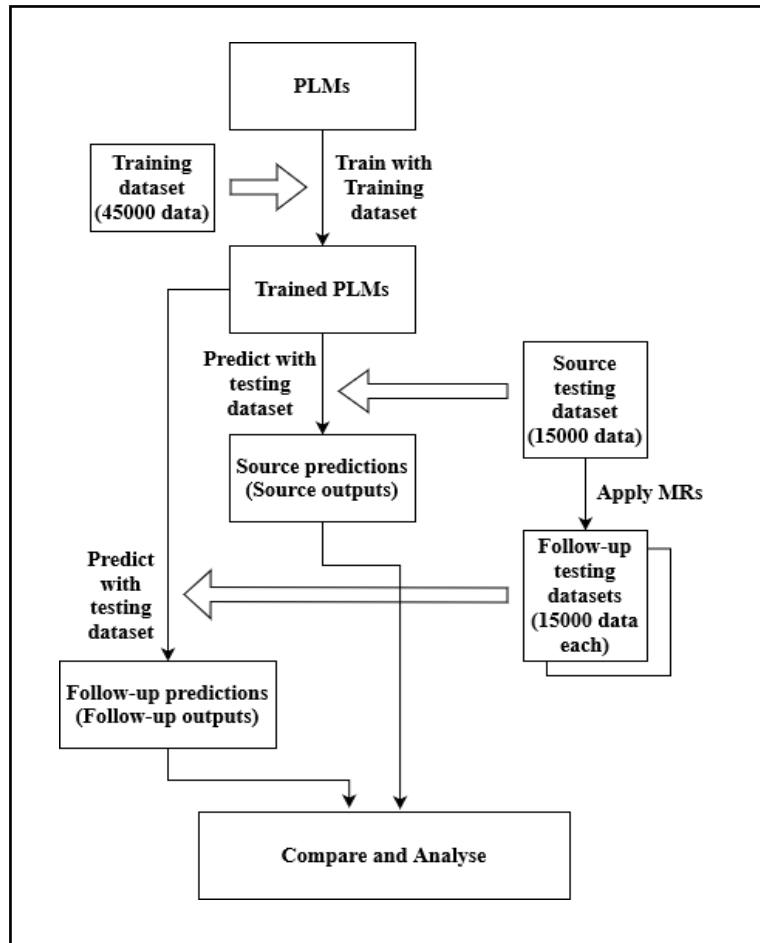


Figure 6.3: Illustration of the experiment (Step 1 to 5).

Step 4. PLMs make predictions on thirteen “Follow-up” testing datasets to create thirteen sets of follow-up outputs.

Step 5. We compare source and follow-up outputs and analyse their differences in measurement. A failure is revealed if the relation is violated, which means differences occur between the outputs (Fig. 6.3).

Step 6. If there is zero violation with the whole group of MRs, a non-violation region is identified for supporting simulation metamorphic testing.

Four trained PLMs were tested with one source testing dataset and thirteen follow-up testing datasets. Thus, this study had four sets of source outputs and fifty-two sets of follow-up outputs for comparison.

6.5.2.4.7 Experimental results and discussion

We first studied the violations of all MRs with all subject PLMs to evaluate the effectiveness of our proposed MRGs (RQ4). Then, we analysed the zero-violation cases to seek an understanding of locating the non-violation region to support simulation MT (RQ5). Finally, we further analysed the results to examine the reasons for the violations.

Table 6.14: RR values of four PLMs with all MRs.

		XLNet	XLM-Roberta	GPT1	T5	Average
MRG1	MR1.1	0.41%	0.37%	0.00%	0.35%	0.28%
	MR1.2	0.66%	0.37%	0.00%	0.37%	0.35%
	MR1.3	0.97%	0.37%	0.00%	0.37%	0.43%
	MR1.4	1.31%	0.37%	0.00%	0.37%	0.51%
	MR1.5	1.57%	0.37%	0.00%	0.37%	0.58%
MRG2	MR2.1	0.02%	0.02%	0.04%	0.04%	0.03%
	MR2.2	0.05%	0.11%	0.39%	0.11%	0.17%
	MR2.3	0.06%	0.11%	0.40%	0.11%	0.17%
	MR2.4	0.06%	0.11%	0.40%	0.14%	0.18%
MRG3	MR3.1	24.30%	19.40%	0.00%	17.85%	15.39%
	MR3.2	26.95%	21.27%	0.00%	20.39%	17.15%
	MR3.3	27.09%	21.33%	0.00%	20.48%	17.23%
	MR3.4	27.11%	21.35%	0.00%	20.46%	17.23%

Zero violation with MRG1. Table 6.14 shows that GPT1 had zero RR with all MRs of MRG1. Although MRG1 only duplicates the content of the title, it is effective MRG as only GPT1 achieves zero violation instead of all PLMs. Thus, MRG1 can be used as a validation tool for content quality classification on Stack Overflow problems using GPT1. As GPT1 considers all MRs in MRG1 to be semantic equivalent, any case that violates any MRs of MRG1 can reveal a fault in the model. Therefore, this result can be interpreted as “duplicating the content of the title from

one time to six times” is the non-violation region. In other words, if GPT1 finds Stack Overflow questions with duplicated content of the title (one time to six times) semantically different, this can be interpreted as a fault in the model without testing. As this region can determine whether the question is violated without testing [37], this region can support simulation metamorphic testing.

Table 6.15: ACC values of four PLMs with all MRs and Source.

		XLNet	XLM-Roberta	GPT1	T5
	SRC	85.67%	85.86%	46.15%	86.15%
MRG1	MR1.1	85.51%	85.72%	46.15%	86.06%
	MR1.2	85.51%	85.72%	46.15%	86.06%
	MR1.3	85.50%	85.72%	46.15%	86.06%
	MR1.4	85.41%	85.72%	46.15%	86.06%
	MR1.5	85.44%	85.72%	46.15%	86.06%
MRG2	MR2.1	85.69%	85.87%	46.16%	86.16%
	MR2.2	85.67%	85.89%	46.20%	86.18%
	MR2.3	85.68%	85.89%	46.20%	86.18%
	MR2.4	85.68%	85.88%	46.20%	86.19%
MRG3	MR3.1	73.70%	77.85%	46.15%	78.37%
	MR3.2	71.57%	76.57%	46.15%	76.85%
	MR3.3	71.47%	76.65%	46.15%	76.89%
	MR3.4	71.43%	76.69%	46.15%	76.93%

Table 6.16: MCC values of four PLMs with all MRs and Source.

		XLNet	XLM-Roberta	GPT1	T5
	SRC	78.55%	78.86%	20.27%	79.22%
MRG1	MR1.1	78.30%	78.63%	20.27%	79.09%
	MR1.2	78.28%	78.63%	20.27%	79.09%
	MR1.3	78.27%	78.63%	20.27%	79.09%
	MR1.4	78.13%	78.63%	20.27%	79.09%
	MR1.5	78.17%	78.63%	20.27%	79.09%
MRG2	MR2.1	78.58%	78.87%	20.28%	79.24%
	MR2.2	78.55%	78.90%	20.32%	79.27%
	MR2.3	78.56%	78.90%	20.32%	79.27%
	MR2.4	78.56%	78.89%	20.32%	79.29%
MRG3	MR3.1	61.22%	67.01%	20.27%	68.95%
	MR3.2	58.70%	65.14%	20.27%	67.60%
	MR3.3	58.56%	65.25%	20.27%	67.79%
	MR3.4	58.51%	65.30%	20.27%	67.85%

Answering RQ4 and RQ5: MRG1 can validate GPT1 in content quality classification on Stack Overflow problems. In addition, as GPT1 did not violate the whole range of MRs in MRG1, this result can create a non-violation region and prove that MRs of MRG1 can support simulation metamorphic testing for GPT1.

Table 6.17: Four extra MRs related to uppercase conversion.

MR group (MRG)	Descriptions
Extra MRG3: Uppercase conversion	MR3.5 – Convert 10% texts to uppercase
	MR3.6 – Convert 37.5% texts to uppercase
	MR3.7 – Convert 62.5% texts to uppercase
	MR3.8 – Convert 87.5% texts to uppercase

Table 6.18: RR values of GPT1 with extra MRs related to uppercase conversion.

	GPT1
MR3.5	0.00%
MR3.6	0.00%
MR3.7	0.00%
MR3.8	0.00%

Zero violation with MRG3. Table 6.14 shows GPT1 has zero RR with all MRs of MRG3. We further tested GPT1 with another set of extra MRs related to uppercase conversion (Table 6.17).

GPT1 also achieved zero violations with the four extra MRs related to MRG3 (Table 6.18). Results show that GPT1 considered all MRs in MRG3 to be semantically equivalent, even though GPT1 is not an uncased version. Thus, MRG3 can be used as a validation tool for the content quality classification on Stack Overflow problems using GPT1.

GPT1 finds all MRs, including extra MRs, in MRG3 to be semantically equivalent, the results indicate no violation from 0% to 100% uppercase conversion. Any case that violated any MRs of MRG3 can reveal a fault in the model. Therefore, this result can be interpreted as “uppercase conversion from 0% to 100% of all texts” is the non-violation region. In other words, if GPT1 finds Stack Overflow questions semantically different between uppercase and lowercase format, this can be

interpreted as a fault in the model without testing. As this region can determine whether the question is violated without testing [37], this region can support simulation metamorphic testing.

Answering RQ4 and RQ5: MRG3 can validate GPT1 in the content quality classification on Stack Overflow problems. In addition, as GPT1 did not violate any MR in MRG3, these results create a non-violation region and prove that MRs of MRG3 can support simulation metamorphic testing for GPT1.

Further analysis

We further analyse with “GPT1 performance” and “Non-zero violation”.

GPT1 performance. Although GPT1 achieves zero violation with MRG1 and MRG3, tables 6.15 and 6.16 show that the ACCs and MCCs of GPT1 are the lowest in the study. It is argued that the uni-directional model, which is GPT1, could be less useful than bi-directional models, such as XLM-Roberta, in making inferences at the sentence level or in learning common sense [71, 149]. It is because uni-directional model, such as GPT1, uses a left-to-right architecture, where every token can only attend to its left context, whereas a bidirectional model incorporates context from both directions [71, 149].

It can be explained by GPT1’s worse performance in RR with MRG2 than with MRG1 and MRG3. MRG1 duplicates the content of the title, and MRG3 changes the case format of the texts; those MRs do not alter the left and right positions of texts in the contexts. On the other hand, MRG2 is expanding abbreviations in the title and body texts to their full forms, and these changes have a more significant impact on the position of texts in the contexts.

For example, GPT1 might find “I don’t eat and don’t drink” and “I do not eat and do not drink” semantically different as based on the left-to-right architecture, tokens “eat” and “drink” might be interpreted differently as the number of left tokens is changed. On the other hand, “I don’t eat” and “I DON’T EAT” or “I don’t eat” and “I don’t eat I don’t eat”, tokens “eat” and “drink” might not be interpreted differently as the changes on the left have a less significant impact on the position of texts in the contexts.

Therefore, it is essential to understand the associated PLMs before selecting appropriate semantic-preserving MRGs for validation and simulation metamorphic testing. This study demonstrates that MRs, which alter the position of texts in the contexts while preserving their semantic meaning, can impact the stability of the performance of GPT1 and other uni-directional models. This makes the abbreviation replacement type of MR (MRG2) unsuitable as a validation tool for uni-directional models.

Non-zero violation. Three PLMs violated all MRs of MRG1, all PLMs violated all MRs of MRG2, and three PLMs violated all MRs of MRG3 (Table 6.14). MRs are supposed to be semantic-preserving; however, the results show that most PLMs do not comply and re-label the data, which indicates that PLMs considered the follow-up cases to be not semantically equivalent to source cases.

Firstly, tables 6.14, 6.15 and 6.16 show that XLNet, XLM-Roberta and T5’s RRs of all MRs of MRG3 are over 17.85%, and their ACCs and MCCs of follow-up outputs are lower than the ACCs and MCCs of source outputs. It can be explained that these three PLMs are case-sensitive. Thus, uppercase conversion is not a suitable validation tool for XLNet, XLM-Roberta and T5.

Secondly, the average RR of MRG1 and MRG2 are less than 1%, and MRG2 has the lowest average RR in the study. However, no PLM achieved zero violation in MRG2. We reviewed the classification report of all PLMs and located the re-label

cases with MRG2 of all PLMs. We further investigated those violated cases and found that all the abbreviation replacements took place in the title text or the first line of the body text. In other words, the position of abbreviation replacement in the context can affect the prediction of PLMs, which agrees with [72] that PLMs use the statistical likelihood of words and patterns to disambiguate the sentence without the need to know the semantic content of the sentences.

These results also agree with [71] that the common sense knowledge of PLMs remains at a surface level instead of deep semantic comprehension, and models are still behind human-level common sense reasoning. As all MRs in this study are semantic-preserving, it is not logical to find contexts semantically different due to the case format, the position of abbreviation replacements or the number of duplications of the title texts. Even though all the PLMs in this study had already trained with a training dataset containing 45000 Stack Overflow questions with quality ratings, the levels of common sense knowledge of PLMs might not be improved due to the limitations of PLMs' capacities [71].

Thirdly, the results show that PLMs use statistical patterns to disambiguate the sentence, so studying those statistical patterns and exploring informative statistical features could be an alternative method to improve the performance of PLMs. [195]'s study demonstrated the potential of incorporating highly informative statistical features, such as word frequency on labels, into PLMs to significantly enhance their performances. Thus, further research is necessary to explore different informative statistical features on different labels and turn them into effective representations to enhance the performance of PLMs.

Therefore, the semantic comprehension and common sense knowledge of PLMs remains at the surface level, so most of the PLMs in the study cannot achieve zero violation with the proposed semantic-preserving MRs. Further investigations on how to improve the levels of semantic comprehension and common sense knowledge of

PLMs, as well as how to explore different informative statistical features and turn them into effective representations, are required to improve their performances.

6.5.2.4.8 Threats to Validity

Internal validity. Internal validity refers to the degree of confidence to which the causal relationship studied in research is independent of other factors, so the correctness of the experimental environment is significant for internal validity. One internal threat to the experiment is parameter configuration, as different input parameters can cause result variations. Standard parameters were applied to all PLMs, and we built PLMs with the pretrained models provided by the machine learning community “Hugging Face” and the libraries of “Python”. We also used Microsoft Excel to create the “Follow-up” testing datasets. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Another concern is the measurements of this study. We focused on the relationship between MRs and measurements, and the prediction outcomes of all PLMs were listed and checked to ensure that the measurements mentioned in section 6.5.2.4.5 were calculated accurately.

External validity. External validity refers to how well the outcome of a study can be expected to apply to other settings, which means how generalizable the findings are. One external threat is the generality of our approach. This study uses semantic-preserving MRs, which are generic and applicable to any experiment using the English language, so the potential threat to the external validity of this study is low. This study chose the “Stack Overflow Questions with Quality Rating” dataset of Kaggle, which has 60000 data, and four different PLMs were employed to run trials with thirteen MRs. Thus, this study had four sets of source outputs and fifty-two sets of follow-up outputs for comparison. We believe the results should be generalised.

At last, although the datasets we used for experiment assessment may vary case by case, the subject PLMs were assessed and evaluated by MT, which is a widely used and general approach. Thus, we argue that the dataset's effect on our approach's effectiveness should be small.

Construct validity. Construct validity concerns the extent to which the test or measure accurately assesses what it is supposed to. The independent variables of this study are prediction outcomes of the “Source” and “Follow-up” testing datasets of four different PLMs, which were trained by the same training dataset. The dependent variable is the differences between the independent variables, which are the differences between the prediction outcomes of “Source” and “Follow-up” testing datasets of PLMs, the RR. As this study focuses on validating PLMs, RR could show the violations. Suppose there is no violation between source prediction outcomes and follow-up prediction outcomes, which means no inconsistent prediction between them. In that case, RR is zero, and the associated PLM is insensitive to the related MR, deeming the MR suitable for validation. Conversely, violating the MR could reveal an error in PLM because the related PLM should be insensitive to the MR. That can correctly measure the intent of this study. To sum up, the construct validity of the independent variables and dependent variables of this study should be acceptable.

6.5.2.4.9 Limitations

This study only experimented on a three-labelled classification problem, which might lead to a high percentage of mapping. Additionally, due to the constraints on computational resources, this study was conducted using the basic versioned PLMs that utilise a minimal number of parameters. Finally, this experiment only selected datasets in the English language as they suit the proposed MRs.

6.5.2.5 Conclusion

This experiment proved that the symmetric MRs are effective validation tools for the content quality classification on Stack Overflow Q&A using PLMs. GPT1 did not violate MRG1 and MRG3. Additionally, the zero violations in the ranges of MRs in MRG1 and MRG3 highlighted the presence of non-violation regions, proving that the proposed MRs can support simulation metamorphic testing. Therefore, if the Stack Overflow question fails in the non-violation region and the non-violation PLM finds the question semantically different, this can be interpreted as a fault in the PLM without testing.

Despite GPT1 did not violate two MRGs, its performance in ACC and MCC ranked the lowest in this experiment. It can be explained by the uni-directional structure of GPT1 [71, 149], so GPT1 only has no violation with those MRs as those MRs do not alter the position of texts in the contexts.

Furthermore, most PLMs cannot achieve zero violations in this study. It can be explained by PLMs that they might use the statistical likelihood of words and patterns, instead of semantic understanding, to perform language tasks [72]. The common sense knowledge of PLMs remains at a surface level instead of deep semantic comprehension, which makes models behind human-level common sense reasoning [71]. Despite training all PLMs in this study with a training dataset containing 45000 Stack Overflow questions with quality ratings, the training did not notably improve the level of common sense of PLMs [71]. Further investigations can focus on improving levels of semantic comprehension and common sense knowledge of PLMs, as well as exploring informative statistical features and transforming them into effective representations, which are required to improve their performances.

6.5.3 Experiment with Symmetric Metaphoric Relations on enhancing Machine common sense of LLM

After the previous two experiments, we could ensure the potential of symmetric MRs as a validation tool for different ML-based systems. In this section, we further our application of symmetric MRs and conducted the third experiment with the novel approach of utilizing symmetric MRs to enhance the machine common sense of LLM. Ensuring high content quality on Question-Answering (Q&A) educational websites is crucial, and multiple studies have shown that LLMs have been employed to maintain content quality across various Q&A websites [7, 49, 50, 59, 60, 70]. However, the performance of LLMs has revealed limitations in deep semantic comprehension and human-level common sense, often relying on statistical likelihood and patterns for sentence understanding tasks [71, 72]. This experiment aims to address these limitations by exploring the use of symmetric MRs to enhance LLM performance in content quality classification, specifically by improving their machine common sense.

We propose a specific symmetric MR and conduct an experiment on three different LLMs to demonstrate the practicality of our approach. The symmetric MR generates a training dataset incorporating common sense by replacing abbreviations with their full forms, aiming to preserve the semantic meaning of sentences and prediction results. We train the LLMs using different datasets to create distinct groups for comparison. These LLMs then make predictions on various testing datasets to evaluate their levels of machine common sense and performance. The results demonstrate that training LLMs with symmetric MR-generated data can significantly improve the machine common sense level and performance of Bert-based LLMs in the content quality classification of Stack Overflow questions compared to regular training data solely. Additionally, the symmetric MR can be employed as a formally derived natural language processing data augmentation (NLPDA) technique to create reliable training or testing datasets. Future studies can

expand on our approach by extending it to other domains related to education technology and exploring additional MRs to enhance machine common sense and further improve the performance of Bert-based LLMs.

6.5.3.1 Background

6.5.3.1.1 Question-Answering (Q&A) websites

Q&A websites play a significant role as self-learning platforms, offering a wide range of user-generated content that spans from basic guidelines to advanced and expert-level answers. These platforms, such as Quora and Stack Overflow, have become immensely popular, particularly in the realm of software-related queries [7, 50, 186]. Maintaining high-quality content on Q&A websites is essential to ensure a positive user experience and encourage active participation from experts in providing valuable answers [7].

6.5.3.1.2 Large language model (LLM)

LLMs have been widely utilised to uphold content quality on various Q&A websites, including medical sites and Stack Overflow platforms [7, 49, 50, 59, 60, 70]. While the performance of LLMs in this context is generally satisfactory, it falls short of matching human performance. As a result, human reviews and evaluations remain necessary to ensure the safe utilization of LLMs [49, 50, 53]. In addition, LLMs' performances showed that they do not possess deep semantic comprehension or human-level common sense, and only use the statistical likelihood of words and patterns to handle the sentence comprehension tasks [71, 72]. Thus, there is still a gap for LLM to reach robust human-level common sense reasoning [71].

6.5.3.1.3 Machine common sense and natural language processing data augmentation

Improving machine common sense can be achieved by either learning from experience or the Web [196]. Learning from experience is similar to learning from data, and theoretically, more training data can improve the performance of machine learning models. Data augmentation is a widely used approach for increasing training data, even in a data scarcity situation [197-199]. However, studies found that Natural Language Processing Data Augmentation (NLPDA) approaches are mostly derived from heuristics instead of formal or theoretically based approaches [197-199]. Thus, creating a training dataset for machine common sense by NLPDA is an oracle problem because it is difficult to distinguish the correct approach from the potentially incorrect approach [200, 201], and the training data might not be trustworthy if they are not formally or theoretical-based derived.

6.5.3.2 Research Question

MT have long been used for test oracle problems of different domains [19, 22-31]. MT addresses the test oracle problem by examining the expected relationships between input-output pairs in consecutive executions of the systems [142]. This experiment attempts to follow [24]’s study and base the concept of symmetry on formally derived MRs as a formally derived NLPDA approach for creating machine common sense training datasets. It demonstrates the practicality of this approach in answering the following research question with a comprehensive scaled experiment.

RQ6: Can symmetry MR enhance the performance of LLMs through enhancing their machine common sense?

6.5.3.3 Proposed metamorphic relation (MR)

In this experiment, we devised a specific symmetric MR inspired by the work of [24] and centred around the concept of symmetry, which posits that predictions should

remain unaffected when the semantic meanings of sentences remain the same. By leveraging this symmetric MR, we aimed to generate a training dataset that would enhance the machine common sense of the LLM by incorporating patterns associated with common sense reasoning.

Specifically, our proposed symmetric MR, denoted as MR1, assumes that the semantic meaning of a sentence remains unchanged even when all abbreviations within the content are replaced with their corresponding full forms. For instance, the sentences “I don’t eat” and “I do not eat” are expected to convey the same meaning to the LLMs. This metamorphic relation primarily focuses on replacing contractions containing “n’t” with their expanded forms, such as substituting “n’t” with “not”.

6.5.3.4 Experimental Setting

6.5.3.4.1 Test Scenario

Inspired by [7], this experiment focuses on enhancing the performance of a currently operating LLM specifically designed for content quality classification on the Stack Overflow Q&A website. Our objective is to augment the capabilities of the LLM by training it with symmetric MR-generated data, with the aim of improving its performance through the enhancement of its machine common sense. By incorporating the principles of symmetric MRs, we seek to address the limitations of the LLM and elevate its ability to effectively classify and maintain the quality of content on the Stack Overflow platform.

6.5.3.4.2 Subject large language model

Three different LLMs are chosen from the ML community “HuggingFace”¹⁴.

BERT is a bi-directional LLM that makes semantic understanding “from the raw text by jointly conditioning on both left and right context in all layers” [149, 202].

¹⁴ Hugging Face website is <https://huggingface.co/>

T5 is an encoder-decoder transfer learning model for natural language processing that converts every language problem into a text-to-text format [191, 192].

GPT1 is an uni-directional language model, and it is the first generative pre-training transformer-based language model created and released by the technology company “OpenAI”, and it uses language modelling on a large corpus with the ability to process long-range dependencies [151, 194].

6.5.3.4.3 Subject Dataset

This study uses a publicly available dataset from Kaggle¹⁵ created to study the content quality classification of Stack Overflow questions [7]. The dataset contains 60000 Stack Overflow questions in English and serves the intent of this study. Questions are classified into three labels. They are “High-quality posts without a single edit” (HQ), “Low-quality posts with a negative score, and multiple community edits. However, they still remain open after those changes” (LQ_EDIT), and “Low-quality posts that were closed by the community without a single edit” (LQ_CLOSE).

45000 labelled Stack Overflow questions are used as the “training dataset”. We then separate the rest of the 15000 labelled data into two datasets. The “first 5000 labelled data” becomes the “extra training dataset” (5K-EX), and the “second 10000 labelled data” becomes this study’s “testing dataset” (10Ktest). At last, we apply symmetric MR, as the NLPDA, into the “training dataset” to create a “common sense training dataset” containing 5000 labelled data (5K-CS)¹⁶ (Fig. 6.4). All datasets are cleansed and set to lowercase before training and testing.

¹⁵ Dataset is downloaded on 23/09/2023 from the link

<https://www.kaggle.com/datasets/imoore/60k-stack-overflow-questionswith-quality-rate>

¹⁶ This study created 5000 common sense training data to match the size of the extra training dataset, in order to reduce the influence of size differences in training datasets. In addition, this study did not replace “ain’t”, “men’t”, “differe’n’t”, “itn’t”, and “Cullen’t” because it is hard to identify the true meaning of those words, and they only related to seven changes out of 45000 data.

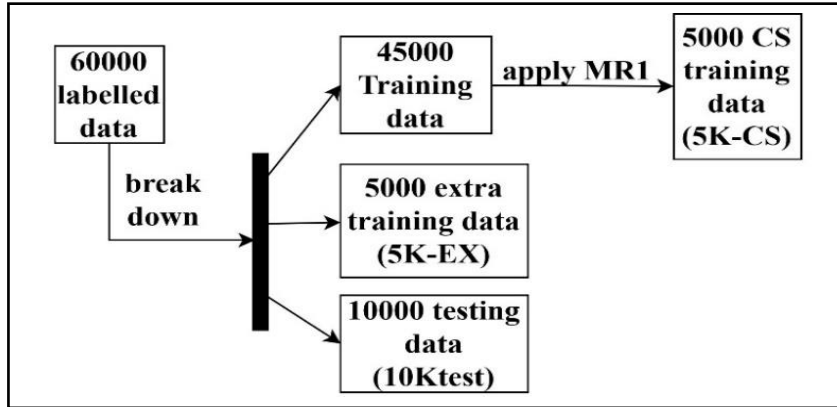


Figure 6.4: Illustration of dataset creations.

6.5.3.4.4 Setup

This experiment environment included using “Intel Core” i7 CPU with 64 GB RAM, “Windows 10” operating systems, “Jupyter Notebook” development platform and “Python” programming language and related libraries, such as “Pytorch”, “transformers”, “tensorflow”, “scikit_learn”, for machine learning algorithms. Standard parameters are applied to all subject PLMs, and the rest of the options are set to default values.

6.5.3.4.5 Measurements

This experiment uses the common ability test created by [71] to evaluate the level of machine common sense and robustness of LLMs. We selected three token-level common sense ability tests (CSAT), and they are “Sense Making” (sm), “Winograd Schema Challenge” (wsc), and “Conjunction Acceptability” (ca). We also selected one sentence-level CSAT, which is “Argument Reasoning Comprehension Task” (arct1). Four robust tests (RT) are chosen, and they are “add”, “del”, “sub”, and “swap” [71]. All the test instances are created by replacing either pronouns, verbs, adjectives, conjunction words or keywords to create dual pairs of statements with labels for models to predict. The scores are calculated by the number of correct predictions the model makes [71].

We also employ the prediction accuracy rate (ACC) and Matthew correlation coefficient (MCC) [73]. These measurements can show the prediction accuracy. Let T_p be true positives that are the actual positives and correctly predicted, T_n be true negatives that are the actual negatives and correctly predicted, F_p be false positives that are the actual negatives and wrongly predicted as positives, and F_n be false negatives are the actual positives and wrongly predicted negatives. ACC is calculated as Equation (6.12), and MCC is calculated as Equation (6.13).

$$ACC = (T_p + T_n) / (T_p + T_n + F_p + F_n) \quad (6.12)$$

$$MCC = \frac{(T_p \times T_n - F_p \times F_n)}{\sqrt{((T_p + F_p) \times (T_p + F_n)) \times ((T_n + F_p) \times (T_n + F_n))}} \quad (6.13)$$

6.5.3.4.6 Implementation

Our experiment includes five steps:

Step 1 - We train the subject LLMs with a 45000 training dataset to create Group 1 (G1) as the currently operating LLMs.

Step 2 - We train LLMs(G1) with 5K-EX to create Group 2 (G2).

Step 3 - We train LLMs(G1) with 5K-CS to create Group 3 (G3) (Fig. 6.5).

Step 4 - LLMs(G1), LLMs(G2) and LLMs(G3) make predictions on 10Ktest and common ability testing datasets.

Step 5 - We compare and analyse all the outputs from Step 4 to evaluate whether there is any improvement (Fig. 6.6).

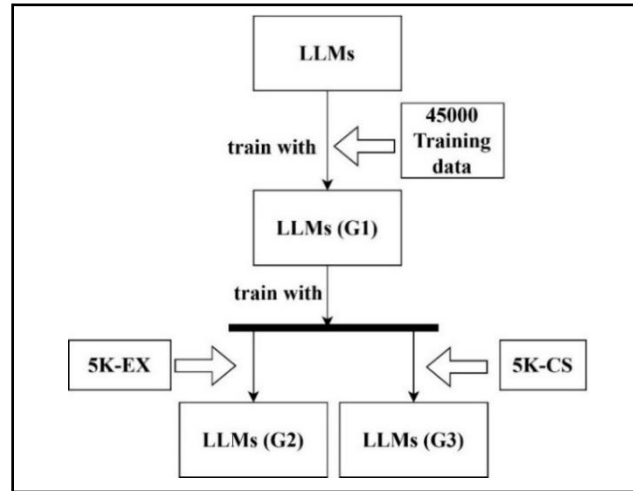


Figure 6.5: Illustration of Step 1 to Step 3.

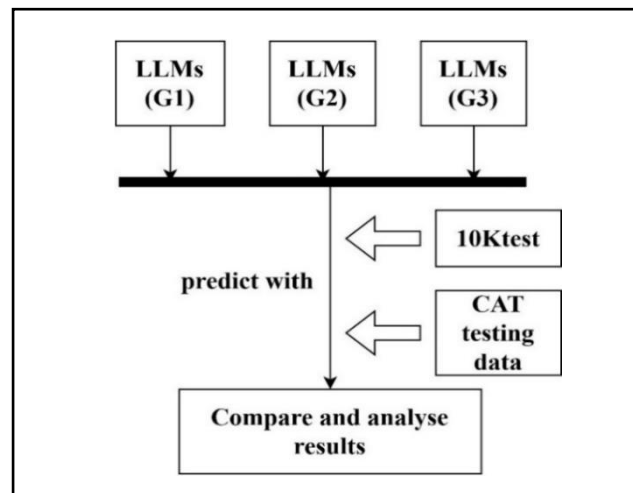


Figure 6.6: Illustration of Step 4 to Step 5.

6.5.3.4.7 Experimental results and discussion

We discussion with “Machine common sense and performance” and further analyse with the findings.

Machine common sense and performance. The results of our study indicate that the proposed approach is most effective when applied to Bert-based LLMs, as

demonstrated in Table 6.19¹⁷. Specifically, Bert(G3) exhibited superior accuracy (ACC) and Matthews correlation coefficient (MCC) compared to Bert(G2). These findings suggest that Bert-based LLMs can benefit significantly from the integration of symmetric MR-generated data, surpassing the performance achieved with regular data alone. In contrast, GPT1-based and T5-based LLMs did not exhibit any improvement in ACC and MCC when subjected to the proposed approach. Notably, GPT1(G1), GPT1(G2), and GPT1(G3) demonstrated identical results in terms of content quality classification accuracy, MCC, CSAT, and RT. Consequently, it is evident that this approach does not yield meaningful enhancements for GPT1-based and T5-based LLMs. Therefore, based on the study's findings, it is clear that the proposed approach is most suitable for enhancing Bert-based LLMs, while it does not provide significant improvements for GPT1-based and T5-based LLMs.

Table 6.19 also shows that “sm”, “wsc”, and “arct1” results are positively related to ACC and MCC, especially “sm” but not “ca”. This can be explained by the fact that LLMs are focusing on the content quality classification of stack overflow questions, so LLMs are required to understand the content of the stack overflow question to make a judgement. “sm”, “wsc” and “arct1” are tests that change the pronouns or nouns of sentences, whereas “ca” changes the conjunction words, which might not be significant enough for content quality classification of stack overflow questions. Thus, enhancing machine common sense that related to “sm”, “wsc”, and “arct1” tests can improve LLMs’ performance in classifying the content quality of stack overflow questions.

However, the RT results are not positively related to ACC and MCC. Table 6.19 shows that despite T5(G3) having better RT results than the other two T5-based LLMs, it has worse ACC and MCC. Similar to Bert(G3) having worse RT results than the other two Bert-based LLMs, it has better prediction results. These RT results are similar to [71] as the subject LLMs might be confused with the modifications in

¹⁷ We omit showing the results of GPT1 in Table 6.19 because they are all identical.

RT; thus, enhancing the robustness related to RT cannot improve the performance in classifying the content quality of stack overflow questions.

Answering RQ6: Symmetric MR can enhance the performance of LLMs through enhancing their machine common sense

Table 6.19: The CSAT, ACC and MCC results of the experiment.

		Bert (G1)	Bert (G2)	Bert (G3)	T5 (G1)	T5 (G2)	T5 (G3)
	ACC	0.8696	0.8525	0.8580	0.8672	0.8649	0.8508
	MCC	0.8040	0.7873	0.7920	0.8008	0.7997	0.7851
CSAT	sm	0.5242	0.5168	0.5136	0.4885	0.4832	0.4832
	wsc	0.4947	0.5053	0.5088	0.4982	0.4947	0.4947
	ca	0.5628	0.5738	0.5464	0.5683	0.5738	0.5574
	arct1	0.4932	0.4910	0.4955	0.4707	0.4662	0.4685
RT	add	0.1630	0.1413	0.1304	0.2826	0.2935	0.2935
	del	0.2561	0.2439	0.2195	0.3171	0.3049	0.3902
	sub	0.1733	0.2267	0.2000	0.2000	0.1733	0.1867
	swap	0.3514	0.4054	0.3514	0.4459	0.4459	0.4054

Further Analysis. The analysis of the results suggests that Bert-based LLMs can effectively utilize symmetric MR-generated training datasets for enhancing their machine common sense and performance. This approach becomes particularly valuable when acquiring additional training data is challenging. Furthermore, employing symmetric MR as a formally derived NLPDA approach, which is based on the concept of symmetry, allows for the creation of trustworthy training and testing datasets.

While BERT(G1) demonstrates the best ACC and MCC results, it aligns with previous research [71] indicating that simply increasing the amount of data may not necessarily lead to improved performance. It is important to note that this study focused on employing a single symmetric MR to enhance machine common sense, and the MR-generated training data proved more effective in improving the

performance of Bert-based LLMs compared to regular training data. These findings provide guidance for future research in the creation of MRs to enhance the machine common sense and performance of Bert-based LLMs. Ultimately, a standardised set of machine common sense MRs can be developed to incorporate various machine common sense patterns into different LLMs to bridge the gap between machines and humans regarding common sense reasoning.

6.5.3.4.8 Threats to Validity

Internal validity. Internal validity refers to the degree of confidence to which the causal relationship studied in research is independent of other factors, so the correctness of the experimental environment is significant for internal validity. One internal threat to the experiment is parameter configuration, as different input parameters can cause result variations. Standard parameters were applied to all PLMs, and we built PLMs with the pretrained models provided by the machine learning community “Hugging Face” and the libraries of “Python”. We also used Microsoft Excel to create the “Follow-up” testing datasets. These tools and technologies should be considered reliable since they have been widely used. White-box testing has also been conducted so the source codes are error-free. Another concern is the measurements of this study. We focused on the relationship between MRs and measurements, and the prediction outcomes of all PLMs were listed and checked to ensure that the measurements mentioned in section 6.5.3.4.5 were calculated accurately.

External validity. External validity refers to how well the outcome of a study can be expected to apply to other settings, which means how generalizable the findings are. One external threat is the generality of our approach. This study uses semantic-preserving MR, which are generic and applicable to any experiment using the English language, so the potential threat to the external validity of this study is low.

This study chose the “Stack Overflow Questions with Quality Rating” dataset of Kaggle, which has 60000 data, and four different PLMs were employed to run trials. We believe the results should be generalised. At last, although the datasets we used for experiment assessment may vary case by case, the subject PLMs were assessed and evaluated by MT, which is a widely used and general approach. Thus, we argue that the dataset's effect on our approach's effectiveness should be small.

Construct validity. Construct validity concerns the extent to which the test or measure accurately assesses what it is supposed to. The independent variables of this study are prediction outcomes of the “Source” and “Follow-up” testing datasets of four different PLMs, which were trained by the same training dataset. The dependent variable is the differences between the independent variables, which are the differences between the prediction outcomes of “Source” and “Follow-up” testing datasets of PLMs, the RR. As this study focuses on machine common sense, CSAT, RT, ACC and MCC could show it. In that case, CSAT, RT, ACC and MCC are all the same between the prediction outcomes of “Source” and “Follow-up”, and the associated PLM is insensitive to the related MR, deeming the MR cannot enhance machine common sense. To sum up, the construct validity of the independent variables and dependent variables of this study should be acceptable.

6.5.3.4.9 Limitation

This study only experimented on a subset of CSAT and RT. Additionally, due to the constraints on computational resources, this study was conducted using the basic versioned PLMs that utilise a small number of parameters. Lastly, this experiment only selected datasets in the English language as they suit the proposed MRs.

6.5.3.5 Conclusion

This experiment has demonstrated that utilizing symmetric MRs to generate training data can significantly enhance the performance of Bert-based LLMs in the content quality classification of Stack Overflow questions and improve their machine common sense, surpassing the performance achieved with regular training data alone. Therefore, symmetric MRs have proven to effectively enhance LLMs' performance by improving their machine common sense. Moreover, applying symmetric MRs as a formally derived NLPDA approach can provide trustworthy training or testing datasets based on the concept of symmetry. This finding opens possibilities for utilizing symmetric MRs in other domains to improve machine common sense and enhance the performance of different LLMs.

6.6 Chapter summary

Chapter 5 brought up an important question about some MRPs that cannot validate LLMs; therefore, in this chapter, we conduct three experiments with the symmetry MR pattern only as a more straightforward generic MT approach for validating different LLMs. The result proved that this is a valuable and applicable validation approach for different LLMs.

In the first experiment, we assessed using symmetric MRs of MT to validate code generation LLMs and to study the effectiveness of MT in validating the stability of LLMs' outputs. The results demonstrated that the proposed MRs could serve as a validation tool with violation measurements for the stability of code generation LLMs' outputs under the simulation that there is no ground truth available. The effectiveness and performance results, which include identifying the inconsistencies between performance, "Source" predictions, "Follow-up" predictions, and ground truths related to MRs, indicated that proposed MRs are indeed effective tools for explaining the instability of LLM's outputs by "Case transformation of prompts", "Duplication of prompts", and "Paraphrasing of prompts". In addition, the results

underscored the importance of enhancing LLMs’ semantic understanding of prompts for better stability. This experiment also highlighted potential future research directions, including exploring different MRs, enhancing semantic understanding, and applying symmetry to prompt engineering.

In the second experiment, we assessed using symmetric MRs of MT to validate PLMs that were utilised in maintaining the content quality of Q&A websites. The results showed that the symmetric MRs could effectively serve as a validation tool for content quality classification on Stack Overflow Q&A using PLMs. One PLM did not violate the “Uppercase conversion” MRG and the “Duplication” MRG. Furthermore, the absence of violations in the MRGs allowed for the identification of non-violation regions, confirming the ability of the proposed MRs to support simulation metamorphic testing. However, further investigations are required to enhance the semantic comprehension and common sense knowledge of PLMs and explore highly informative statistical patterns of PLMs, in order to improve their overall performance.

In the third experiment, we assess the potential practicality of using symmetric MRs as a formally derived NLPDA approach, which can provide trustworthy training or testing datasets, to enhance LLMs’ performance by improving their machine common sense. The experiment findings demonstrate that a Bert-based LLM trained with symmetric MR-generated data outperforms a Bert-based LLM trained solely with regular labelled data. This outcome highlights the effectiveness of symmetric MRs in enhancing LLMs’ performance by improving their machine common sense. Subsequent studies can extend our approach to other domains to improve machine common sense and enhance the performance of different LLMs.

In conclusion, the symmetric MRs have been demonstrated to be a valuable and straightforward validation approach applicable to various ML-based systems, fulfilling our main objective to create a simple and universal generic validation tool for various LLMs. Furthermore, this approach can be utilised in explaining the

instability of LLM's outputs, supporting simulation MT and as a formally derived NLPDA approach. However, all three experiments only selected datasets in the English language as they suit the proposed MRs. The next chapter will conclude this thesis and give future directions.

Chapter 7. Conclusion and Future Work

This thesis has been devoted to the study of validating machine learning-based (ML-based) systems by using different novel generic metamorphic relations (MRs) of metamorphic testing (MT) approaches. This chapter summarises our main findings and contributions and discusses promising future directions.

7.1 Summary

In this thesis, we proposed using generic MRs of MT approach for validating ML-based systems, and we experimented with different ML-based educational systems to address Problem 1 (Difficulty in identifying Metamorphic Relations of Metamorphic Testing) and Problem 2 (Lack of awareness of the need to research validation approach of machine learning-based educational systems). Through a series of experiments, we evaluated the effectiveness of generic MRs of MT approach as a validation tool for ML-based systems.

In Chapter 3, we conduct an experiment that adapts the existing MT model into a generic MT model for validating ML-based software defect prediction systems (SDPs). The results demonstrated that the proposed generic MT model is a valuable validation tool for SDPs. However, this model can only be applied to features with

numeric values. Therefore, we shifted our focus to uncertainty in order to explore different generic MT approaches.

In Chapter 4, we conduct an experiment that derives generic MRs of MT from an uncertainty matrix for validating ML-based plagiarism detection systems (PDSs). The findings show that uncertainty-based generic MT can also be a validation tool for ML-based systems. However, this approach, even when utilizing the uncertainty matrix, relies on uncertainties that may require input from experts or users to identify, potentially limiting its generality accurately. Consequently, we shift our focus to metamorphic relation patterns (MRPs).

In Chapter 5, we conduct an experiment that derives generic MRs of MT from MRPs for validating ML-based PDSs. The result proved that MRP-based generic MT can serve as a validation tool for ML-based systems, and it is indeed generic. Thus, we see the potential of MRPs in developing generic MRs, and we shift our focus on experimenting with the symmetry MRP to develop a more straightforward and applicable generic MT approach for validation in the next chapter.

In Chapter 6, we conduct three experiments with symmetric MRs derived from the symmetry MRP as a more straightforward generic MT approach for validating various ML-based systems. The results demonstrate that this approach is both valuable and applicable validation approach for different ML-based systems. The symmetry MRP has proved to be an effective, uncomplicated, and widely applicable validation technique. Additionally, it can elucidate the instability of LLM outputs, facilitate simulation MT, and function as a formally derived Natural Language Processing Data Augmentation (NLPDA) approach.

The findings presented in this thesis demonstrate that the use of generic MRs of MT approach serves as a novel and valuable validation tool for various types of ML-based systems. Additionally, this approach can be applied across different areas. We hope that our encouraging results will inspire further research in this field, ultimately leading to more reliable ML-based systems through our proposed methods.

7.2 Contributions

The main contributions resulting from the empirical investigations carried out in this thesis are listed below.

Firstly, in the area of validating ML-based systems, especially the lack of research awareness area: PDSs and other educational technologies, we propose different novel approaches to deriving generic MRs of MT for validation without the need for systems users' or experts' inputs and experiment with them and different ML-based educational systems.

Secondly, this thesis has demonstrated that the generic MRs of MT can be extended to validate various domains using different languages and ML algorithms, including Large Language Models (LLMs). To the best of our knowledge, we are the first to utilise generic MRs of MT, which are based on adapted generic MT model, uncertainty and metamorphic relation patterns (MRPs), for validating different ML-based systems, such as code-generation systems (CGs), content quality classification systems (CQCs), software defect prediction systems (SDPs) and PDSs.

Thirdly, this thesis has developed different systematic and effective approaches to derive generic MRs for researchers and practitioners to follow and apply, even without prior knowledge of the validating targets.

Fourthly, we discovered that symmetric MRs derived from symmetry MRP can be utilised not only as a general validation tool for different ML-based systems. Symmetric MRs can also be used as a formal “Natural Language Processing Data Augmentation” (NLPDA) approach to provide trustworthy training or testing datasets, to create non-violation regions to support simulation MT, and to explain the instability of ML-based systems' outputs. These findings open up possibilities for utilising symmetric MRs to further research on improving machine common sense of systems, simulation MT in different domains, and validating and identifying the causes of the instabilities with other domains using different languages and LLMs.

Fifthly, we experimented with applying Explainable Artificial Intelligence (XAI) to analyse the results for internal validity. This application opens up the possibility of future research applying XAI to validate their findings.

Sixthly, we have outlined several research challenges and opportunities that researchers may encounter in extending our proposed approaches for machine learning-based systems in other areas.

7.3 Future Works

This thesis confirms our hypothesis of validating ML-based systems with generic MT, and it opens several directions for future research.

7.3.1 Further explore and investigate Symmetry MRP

We discovered that symmetric MRs, derived from symmetry MRP, can serve as a valuable validation tool for various ML-based systems. Future research could replicate this study across domains that employ various ML algorithms to explore how this pattern can function as a validation method for diverse ML algorithms. Ultimately, this approach could become the standard validation method for most ML-related research. In addition, the proposed symmetric MRs used syntactic changes, format changes and paraphrasing to preserve the semantic meanings, which are very conservative. It is highly suggested that more variations be experimented with on creating symmetric MRs, especially more specific symmetric MR creations, to pinpoint the areas that can more precisely validate the systems under investigation.

7.3.2 Explore and investigate other generic MT approaches

Future research can replicate the experiments conducted in this thesis across various domains to investigate the relationships between configurations of ML algorithms, characteristics of ML algorithms, and MRs and how to establish effective generic

MRs for validating diverse ML algorithms. Additional studies are necessary to compare our methods with other state-of-the-art traditional and generic MT approaches, in order to identify potential improvements. Furthermore, future studies may enhance our MT model by incorporating different effectiveness measurements for validation across different domains, such as expanding our code-generation experiment to include class-level or line-level code generation LLMs and other code-generation testing datasets using different programming languages, such as Java.

7.3.3 Explore and investigate enhancing machine common sense

Applying symmetry MRP to derive MRs as a formally derived NLPDA approach can provide trustworthy training or testing datasets. This finding opens up possibilities for utilizing symmetric MRs across various domains, thereby enhancing machine common sense and improving the performance of diverse ML-based systems. To further advance the field, future research should build upon our methods by investigating more generic MRs with symmetry and other MRPs, and by extending these concepts to various domains. By leveraging these techniques, we can pave the way for advancements in the performance and reliability of ML-based systems and ultimately work towards developing a standardised set of machine common sense MRs. These attempts will help bridge the gap between machines and humans regarding common sense reasoning.

7.3.4 Better understanding the machine learning-based systems

Another valuable research focus is exploring highly informative statistical features as additional information to ML-based systems and experimenting with symmetric MRs or different MRs. This investigation could assist in better understanding ML-based systems' performances and discovering more suitable MRs for validation tools.

More studies are needed to explore more different MRs with symmetry, which is helpful as different symmetric MRs can generate more testing datasets to measure the effectiveness of validations and further understand the ML-based systems.

This study only focused on the symmetry MRP for validating and understanding ML-based systems. We strongly recommend that future studies explore the “change direction” MRP [24], which is similar to the proven MRP “Partition difference” in Chapter 5, for validating and understanding ML-based further and, at the same time, to explore more MRPs for generic MT. Additionally, exploring other MRPs, such as “conjunctive conditions”, “disjunctive conditions”, “disjoint partitions”, and “complete partitions”, will be another good direction in the future. These research focuses can help to pinpoint the areas that can more precisely validate the LLMs or systems under investigation. However, it is important to note that some MRPs, such as “shuffling” MRP, may not be suitable for validating LLMs as shuffling the words in a prompt can lead to unpredictable results from LLMs. Thus, this research focus must be executed with a better understanding of the subjects.

7.3.5 Explore and investigate other approaches to enhance the performance of machine learning-based systems

Another focus is exploring approaches to turn existing projects into training data systematically for LLMs to improve the semantic understanding of prompts, especially for the prompts of future unseen projects.

Apart from increasing training data, future studies can focus on other approaches, such as mixing different LLMs’ technologies together to improve the semantic understanding of private LLMs and enhance their performance.

Exploring the application of the concept of symmetry to prompt engineering for developing a verifiable, validated, and replicable code generation process could be a potential future research focus.

7.3.6 Explore and investigate other approaches to improve proposed approaches

Due to the limited computer resources, this thesis used datasets that are small in number of instances and features; the proposed model might not be tested thoroughly because it is not difficult for ML-based systems not to violate MR with small datasets. Further studies can redo the experiments with datasets with much higher features and instances to further evaluate the robustness and practicality of the proposed models. In addition, limited computer resources led the experiments to use ML models that were less advanced and contained fewer parameters, thus redoing the experiment with the advanced versions of the chosen technologies, such as applying our experiment in Chapter with the latest version of ChatGPT and Claude to examine the extendibility of our approaches.

Apart from the above, exploring approaches to systematically turn existing projects into training data might improve ML-based systems' performances. Future studies can explore this area, especially for code-generation LLMs, to train with data from existing projects and examine their performance in code-generation on unseen projects.

At last, this thesis mainly focuses on using datasets in English and a limited variety of word embedding approaches. Future studies could consider applying our proposed approaches to different languages and a broader set of embedding approaches to examine the extendibility of the proposed approaches.

Bibliography

- [1] X. Zhang, F. T. Chan, C. Yan, and I. Bose, "Towards risk-aware artificial intelligence and machine learning systems: An overview," *Decision Support Systems*, p. 113800, 2022.
- [2] T. Foltýnek, N. Meuschke, and B. Gipp, "Academic plagiarism detection: a systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1-42, 2019.
- [3] J. Xu, J. Ai, J. Liu, and T. Shi, "ACGDP: An Augmented Code Graph-Based System for Software Defect Prediction," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 850-864, 2022, doi: 10.1109/TR.2022.3161581.
- [4] H. Wang, W. Zhuang, and X. Zhang, "Software defect prediction based on gated hierarchical LSTMs," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 711-727, 2021.
- [5] N. Li, M. Shepperd, and Y. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," *Information and Software Technology*, vol. 122, p. 106287, 2020/06/01/ 2020, doi: <https://doi.org/10.1016/j.infsof.2020.106287>.
- [6] X. Chen, D. Zhang, Y. Zhao, Z. Cui, and C. Ni, "Software defect number prediction: Unsupervised vs supervised methods," *Information and Software Technology*, vol. 106, pp. 161-181, 2019/02/01/ 2019, doi: <https://doi.org/10.1016/j.infsof.2018.10.003>.
- [7] I. Annamoradnejad, J. Habibi, and M. Fazli, "Multi-view approach to suggest moderation actions in community question answering sites," *Information Sciences*, vol. 600, pp. 144-154, 2022.

- [8] F. Cassano *et al.*, "MultiPL-E: A Scalable and Polyglot Approach to Benchmarking Neural Code Generation," *IEEE Transactions on Software Engineering*, vol. 49, no. 7, pp. 3675-3691, 2023.
- [9] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn, "A systematic evaluation of large language models of code," in *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 2022, pp. 1-10.
- [10] J. Austin *et al.*, "Program synthesis with large language models," *arXiv preprint arXiv:2108.07732*, 2021.
- [11] M. Chen *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.
- [12] S. Mouselinos, M. Malinowski, and H. Michalewski, "A Simple, Yet Effective Approach to Finding Biases in Code Generation," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 11299-11329.
- [13] Z. Li *et al.*, "Cctest: Testing and repairing code completion systems," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023: IEEE, pp. 1238-1250.
- [14] M. Yan, J. Chen, J. M. Zhang, X. Cao, C. Yang, and M. Harman, "Coco: Testing code generation systems via concretized instructions," *arXiv preprint arXiv:2308.13319*, 2023.
- [15] Z. Zeng, H. Tan, H. Zhang, J. Li, Y. Zhang, and L. Zhang, "An extensive study on pre-trained models for program understanding and generation," in *Proceedings of the 31st ACM SIGSOFT international symposium on software testing and analysis*, 2022, pp. 39-51.
- [16] G. Poesia *et al.*, "Synchromesh: Reliable code generation from pre-trained language models," *arXiv preprint arXiv:2201.11227*, 2022.

- [17] B. Athiwaratkun *et al.*, "Multi-lingual evaluation of code generation models," *arXiv preprint arXiv:2210.14868*, 2022.
- [18] S. Wang *et al.*, "Recode: Robustness evaluation of code generation models," *arXiv preprint arXiv:2212.10264*, 2022.
- [19] X. Xie, Z. Zhang, T. Y. Chen, Y. Liu, P.-L. Poon, and B. Xu, "METTLE: A metamorphic testing approach to assessing and validating unsupervised machine learning systems," *IEEE Transactions on Reliability*, vol. 69, no. 4, pp. 1293-1322, 2020.
- [20] D. Xiao, Z. LIU, Y. Yuan, Q. Pang, and S. Wang, "Metamorphic Testing of Deep Learning Compilers," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 1, p. Article 15, 2022, doi: 10.1145/3508035.
- [21] C.-a. Sun, B. Liu, A. Fu, Y. Liu, and H. Liu, "Path-directed source test case generation and prioritization in metamorphic testing," *Journal of Systems and Software*, vol. 183, p. 111091, 2022/01/01/ 2022, doi: <https://doi.org/10.1016/j.jss.2021.111091>.
- [22] Z. Ying, D. Towey, A. Bellotti, Z. Q. Zhou, and T. Y. Chen, "Preparing SQA Professionals: Metamorphic Relation Patterns, Exploration, and Testing for Big Data," *Proceedings of the International Conference on Open and Innovation Education (ICOIE 2021)*, pp. 22-30, 2021.
- [23] M. Zhang, J. W. Keung, T. Y. Chen, and Y. Xiao, "Validating class integration test order generation systems with Metamorphic Testing," *Information and Software Technology*, vol. 132, p. 106507, 2021.
- [24] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, "Metamorphic Relations for Enhancing System Understanding and Use," *IEEE Transactions on Software Engineering*, vol. 46, no. 10, pp. 1120-1154, 2020, doi: 10.1109/TSE.2018.2876433.

- [25] B. Stacy, J. Hauzel, M. Lindvall, A. Porter, and M. Pop, "Metamorphic Testing in Bioinformatics Software: A Case Study on Metagenomic Assembly," in *2022 IEEE/ACM 7th International Workshop on Metamorphic Testing (MET)*, 2022: IEEE, pp. 31-33.
- [26] J. D. Ellis, R. Iqbal, and K. Yoshimatsu, "Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing," *Journal of Computational Science*, vol. 55, p. 101456, 2021.
- [27] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: a systematic mapping," *Empirical Software Engineering*, vol. 25, no. 6, pp. 5193-5254, 2020.
- [28] P. Saha and U. Kanewala, "Fault Detection Effectiveness of Metamorphic Relations Developed for Testing Supervised Classifiers," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 4-9 April 2019 2019, pp. 157-164, doi: 10.1109/AITest.2019.00019.
- [29] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *J SYST SOFTWARE*, vol. 84, no. 4, pp. 544-558, 2011, doi: 10.1016/j.jss.2010.11.920.
- [30] S. Pugh, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Systematic testing of post-quantum cryptographic implementations using metamorphic testing," in *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*, 2019: IEEE, pp. 2-8.
- [31] S. Segura, A. Durán, J. Troya, and A. Ruiz-Cortés, "Metamorphic relation patterns for query-based systems," in *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*, 2019: IEEE, pp. 24-31.

- [32] Y. Deng *et al.*, "A Declarative Metamorphic Testing Framework for Autonomous Driving," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1964-1982, 2023.
- [33] Q.-H. Luu, H. Liu, T. Y. Chen, and H. L. Vu, "A Sequential Metamorphic Testing Framework for Understanding Autonomous Vehicle's Decisions," *IEEE Transactions on Intelligent Vehicles*, 2024.
- [34] L. Miao, D. Towey, Y. Ma, T. Y. Chen, and Z. Q. Zhou, "Exploring Metamorphic Testing for Fake-News Detection Software: A Case Study," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2023: IEEE, pp. 912-923.
- [35] L. Xu, D. Towey, A. P. French, S. Benford, Z. Q. Zhou, and T. Y. Chen, "Using metamorphic relations to verify and enhance artcode classification," *Journal of Systems and Software*, vol. 182, p. 111060, 2021.
- [36] Q.-H. Luu, M. F. Lau, S. P. Ng, and T. Y. Chen, "Testing multiple linear regression systems with metamorphic testing," *Journal of Systems and Software*, vol. 182, p. 111062, 2021.
- [37] Z. Ying, A. Bellotti, D. Towey, T. Y. Chen, and Z. Q. Zhou, "Using Metamorphic Relation Violation Regions to Support a Simulation Framework for the Process of Metamorphic Testing," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022: IEEE, pp. 1722-1727.
- [38] M. Jiang, T. Y. Chen, and S. Wang, "On the effectiveness of testing sentiment analysis systems with metamorphic testing," *Information and Software Technology*, vol. 150, p. 106966, 2022.

- [39] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic Testing: A New Approach for Generating Next Test Cases," *arXiv.org*, 2020, doi: 10.48550/arxiv.2002.12543.
- [40] T. Y. Chen *et al.*, "Metamorphic Testing: A Review of Challenges and Opportunities," *ACM Comput. Surv.*, vol. 51, no. 1, p. Article 4, 2018, doi: 10.1145/3143561.
- [41] C. a. Sun, Y. Liu, Z. Wang, and W. K. Chan, " μ MT: A Data Mutation Directed Metamorphic Relation Acquisition Methodology," in *2016 IEEE/ACM 1st International Workshop on Metamorphic Testing (MET)*, 16-16 May 2016 2016, pp. 12-18.
- [42] A. Kumar, R. D. Gopal, R. Shankar, and K. H. Tan, "Fraudulent review detection model focusing on emotional expressions and explicit aspects: investigating the potential of feature engineering," *Decision Support Systems*, vol. 155, p. 113728, 2022.
- [43] I. Setha and H. Aliane, "Enhancing automatic plagiarism detection: Using Doc2vec model," in *2022 International Conference on Advanced Aspects of Software Engineering (ICAASE)*, 2022: IEEE, pp. 1-5.
- [44] R. Wang and Y. Shi, "Research on application of article recommendation algorithm based on Word2Vec and TFIDF," in *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, 2022: IEEE, pp. 454-457.
- [45] C. Zhao *et al.*, "An Improved Term Frequency-Inverse Document Frequency Method Solving Multi-Text Label Problem," in *2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT)*, 2022: IEEE, pp. 400-404.

- [46] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," *IEEE Computational intelligence magazine*, vol. 9, no. 2, pp. 48-57, 2014.
- [47] V. dos Santos *et al.*, "Using Natural Language Processing to Build Graphical Abstracts to be used in Studies Selection Activity in Secondary Studies," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2021: IEEE, pp. 1-8.
- [48] R. Sonbol, G. Rebdawi, and N. Ghneim, "The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review," *arXiv preprint arXiv:2206.00421*, 2022.
- [49] K. Singhal *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172-180, 2023.
- [50] B. Xu *et al.*, "Are we ready to embrace generative AI for software Q&A?," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2023: IEEE, pp. 1713-1717.
- [51] P. Verma. "A professor accused his class of using ChatGPT, putting diplomas in jeopardy." *Washington Post*. <https://www.washingtonpost.com/technology/2023/05/18/texas-professor-threatened-fail-class-chatgpt-cheating/> (accessed 2023-05-19, 2023).
- [52] Y. Xiao, S. Chatterjee, and E. Gehringer, "A New Era of Plagiarism the Danger of Cheating Using AI," in *2022 20th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2022: IEEE, pp. 1-6.
- [53] Y. Chang *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1-45, 2024.

- [54] T. Foltýnek *et al.*, "Testing of support tools for plagiarism detection," *International Journal of Educational Technology in Higher Education*, vol. 17, no. 1, pp. 1-31, 2020.
- [55] I. Albluwi, "Plagiarism in programming assessments: a systematic review," *ACM Transactions on Computing Education (TOCE)*, vol. 20, no. 1, pp. 1-28, 2019.
- [56] R. Kumar, A. Chaturvedi, and L. Kailasam, "An Unsupervised Software Fault Prediction Approach Using Threshold Derivation," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 911-932, 2022, doi: 10.1109/TR.2022.3151125.
- [57] Z. Wan, X. Xia, A. E. Hassan, D. Lo, J. Yin, and X. Yang, "Perceptions, Expectations, and Challenges in Defect Prediction," *IEEE Transactions on Software Engineering*, vol. 46, no. 11, pp. 1241-1266, 2020, doi: 10.1109/TSE.2018.2877678.
- [58] S. Gao, L. Gao, Q. Li, and J. Xu, "Application of large language model in intelligent q&a of digital government," in *Proceedings of the 2023 2nd International Conference on Networks, Communications and Information Technology*, 2023, pp. 24-27.
- [59] R. Mousavi, T. Raghu, and K. Frey, "Harnessing artificial intelligence to improve the quality of answers in online question-answering health forums," *Journal of Management Information Systems*, vol. 37, no. 4, pp. 1073-1098, 2020.
- [60] A. Wen, M. Y. Elwazir, S. Moon, and J. Fan, "Adapting and evaluating a deep learning language model for clinical why-question answering," *JAMIA open*, vol. 3, no. 1, pp. 16-20, 2020.

- [61] W. Zhi, "Multi-modal Medical Q&A System," in *Proceedings of the 2024 International Conference on Computer and Multimedia Technology*, 2024, pp. 414-424.
- [62] Z. Zhang and J. Yang, "HintMiner: Automatic Question Hints Mining From Q&A Web Posts with Language Model via Self-Supervised Learning," in *International Conference on Artificial Intelligence and Statistics*, 2024: PMLR, pp. 271-279.
- [63] A. Kamienski, A. Hindle, and C.-P. Bezemer, "Analyzing techniques for duplicate question detection on Q&A websites for game developers," *Empirical Software Engineering*, vol. 28, no. 1, p. 17, 2023.
- [64] O. Asare, M. Nagappan, and N. Asokan, "Is github's copilot as bad as humans at introducing vulnerabilities in code?," *Empirical Software Engineering*, vol. 28, no. 6, p. 129, 2023.
- [65] S. Troshin and N. Chirkova, "Probing Pretrained Models of Source Codes," in *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2022, pp. 371-383.
- [66] E. Sreedevi, V. PremaLatha, S. Sivakumar, and S. R. Nayak, "A Comparative Study on New Classification Algorithm using NASA MDP Datasets for Software Defect Detection," in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, 21-22 Feb. 2019 2019, pp. 312-317, doi: 10.1109/ISS1.2019.8908096.
- [67] M. J. Hemández-Molinos, Á. J. Sánchez-García, and R. E. Barrientos-Martínez, "Classification Algorithms for Software Defect Prediction: A Systematic Literature Review," in *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2021: IEEE, pp. 189-196.

- [68] S. C. M. Post. "ChatGPT usage could open companies to leaks of corporate secrets and lawsuits, cyber firm warns." South China Morning Post. <https://www.scmp.com/tech/big-tech/article/3217571/chatgpt-usage-could-open-companies-leaks-corporate-secrets-and-lawsuits-cyber-firm-warns> (accessed 24/12/2023, 2023).
- [69] Z. Zhang *et al.*, "A survey on language models for code," *arXiv preprint arXiv:2311.07989*, 2023.
- [70] B. Sen, N. Gopal, and X. Xue, "Support-BERT: predicting quality of question-answer pairs in MSDN using deep bidirectional transformer," *arXiv preprint arXiv:2005.08294*, 2020.
- [71] X. Zhou, Y. Zhang, L. Cui, and D. Huang, "Evaluating commonsense in pre-trained language models," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, no. 05, pp. 9733-9740.
- [72] J. Browning and Y. LeCun, "Language, common sense, and the Winograd schema challenge," *Artificial Intelligence*, p. 104031, 2023.
- [73] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 6-6, 2020, doi: 10.1186/s12864-019-6413-7.
- [74] Y. Hussain, Z. Huang, Y. Zhou, and S. Wang, "Boosting source code suggestion with self-supervised Transformer Gated Highway," *Journal of Systems and Software*, vol. 196, p. 111553, 2023.
- [75] K. K. A. Chua, D.-H. Bae, and E. Jee, "Metamorphic testing for reliability in system of systems," in *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, 2021: IEEE, pp. 390-400.

- [76] C.-a. Sun, H. Jin, A. Fu, Z. Wang, and W. Chan, "Identifying Metamorphic Relations: A Data Mutation Directed Approach," *Available at SSRN 4040492*, 2022. [Online]. Available: <https://ssrn.com/abstract=4040492>.
<https://ssrn.com/abstract=4040492>.
- [77] Z. Xu *et al.*, "A comprehensive comparative study of clustering-based unsupervised defect prediction models," *Journal of Systems and Software*, vol. 172, p. 110862, 2021/02/01/ 2021, doi: <https://doi.org/10.1016/j.jss.2020.110862>.
- [78] F. U. Rehman and C. Izurieta, "An Approach For Verifying And Validating Clustering Based Anomaly Detection Systems Using Metamorphic Testing," in *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 15-18 Aug. 2022 2022, pp. 12-18, doi: 10.1109/AITest55621.2022.00011.
- [79] F. U. Rehman and C. Izurieta, "MT4UML: Metamorphic Testing for Unsupervised Machine Learning," in *2022 9th Swiss Conference on Data Science (SDS)*, 22-23 June 2022 2022, pp. 26-32, doi: 10.1109/SDS54800.2022.00012.
- [80] S. Herbold and T. Haar, "Smoke testing for machine learning: simple tests to discover severe bugs," *Empirical Software Engineering*, vol. 27, no. 2, p. 45, 2022/01/21 2022, doi: 10.1007/s10664-021-10073-7.
- [81] I. A. Venkatkumar and S. J. K. Shardaben, "Comparative study of data mining clustering algorithms," in *2016 International Conference on Data Science and Engineering (ICDSE)*, 2016: IEEE, pp. 1-7.
- [82] E. Patel and D. S. Kushwaha, "Clustering cloud workloads: K-means vs gaussian mixture model," *Procedia Computer Science*, vol. 171, pp. 158-167, 2020.

- [83] SciKit-Learn. "2.1. Gaussian mixture models." SciKit-Learn. <https://scikit-learn.org/stable/modules/mixture.html#gmm> (accessed 2022).
- [84] SciKit-Learn. "sklearn.mixture.GaussianMixture." <https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html> (accessed 2022).
- [85] H. Attias, "A variational bayesian framework for graphical models," *Advances in neural information processing systems*, vol. 12, 1999.
- [86] SciKit-Learn. "sklearn.mixture.BayesianGaussianMixture." <https://scikit-learn.org/stable/modules/generated/sklearn.mixture.BayesianGaussianMixture.html> (accessed 2022).
- [87] SciKit-Learn. "sklearn.cluster.KMeans." SciKit-Learn. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> (accessed 2022).
- [88] J. Peksa, "An Automated Algorithm Implementation to Fill Missing Points with Euclidean Approach," in *2021 4th International Conference on Information and Computer Technologies (ICICT)*, 11-14 March 2021 2021, pp. 79-83, doi: 10.1109/ICICT52872.2021.00020.
- [89] V. Rohilla, S. Chakraborty, and M. S. Singh, "Data clustering using bisecting k-means," in *2019 international conference on computing, communication, and intelligent systems (ICCCIS)*, 2019: IEEE, pp. 80-83.
- [90] SciKit-Learn. "sklearn.cluster.BisectingKMeans." <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.BisectingKMeans.html> (accessed 2022).
- [91] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, pp. 165-193, 2015.

- [92] H. He, Y. He, F. Wang, and W. Zhu, "Improved K-means algorithm for clustering non-spherical data," *Expert Systems*, vol. 39, no. 9, p. e13062, 2022.
- [93] K. Zhang and X. Gu, "An affinity propagation clustering algorithm for mixed numeric and categorical datasets," *Mathematical Problems in Engineering*, vol. 2014, 2014.
- [94] P. Bhattacharjee and P. Mitra, "A survey of density based clustering algorithms," *Frontiers of Computer Science*, vol. 15, pp. 1-27, 2021.
- [95] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," presented at the Proceedings of the 1999 ACM SIGMOD international conference on Management of data, Philadelphia, Pennsylvania, USA, 1999. [Online]. Available: <https://doi.org/10.1145/304182.304187>.
- [96] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Current Software Defect Prediction: A Systematic Review," in *2022 Applied Informatics International Conference (AiIC)*, 2022: IEEE, pp. 117-121.
- [97] L. H. Son, N. Pritam, M. Khari, R. Kumar, P. T. M. Phuong, and P. H. Thong, "Empirical study of software defect prediction: A systematic mapping," *Symmetry*, vol. 11, no. 2, p. 212, 2019.
- [98] H. Chen, Z. Zhang, W. Yin, C. Zhao, F. Wang, and Y. Li, "A study on depth classification of defects by machine learning based on hyper-parameter search," *Measurement*, vol. 189, p. 110660, 2022.
- [99] M. T. Sathe and A. C. Adamuthe, "Comparative Study of Supervised Algorithms for Prediction of Students' Performance," *International Journal of Modern Education & Computer Science*, vol. 13, no. 1, 2021.
- [100] L. Jiang, Z. Cai, D. Wang, and S. Jiang, "Survey of improving k-nearest-neighbor for classification," in *Fourth international conference on fuzzy*

- systems and knowledge discovery (FSKD 2007)*, 2007, vol. 1: IEEE, pp. 679-683.
- [101] S. Karim, H. L. H. S. Warnars, F. L. Gaol, E. Abdurachman, and B. Soewito, "Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset," in *2017 IEEE international conference on cybernetics and computational intelligence (CyberneticsCom)*, 2017: IEEE, pp. 19-23.
- [102] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.
- [103] D. Papakyriakou and I. Barbounakis, "Data Mining Methods: A Review," *International Journal of Computer Applications*, vol. 183, no. 48, pp. 5-19, 01/18 2022, doi: 10.5120/ijca2022921884.
- [104] SciKit-Learn. "sklearn.cluster.AffinityPropagation." <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html> (accessed 2022).
- [105] X. Liu, M. Yin, J. Luo, and W. Chen, "An improved affinity propagation clustering algorithm for large-scale data sets," in *2013 Ninth International Conference on Natural Computation (ICNC)*, 2013: IEEE, pp. 894-899.
- [106] A. M. Serdah and W. M. Ashour, "Clustering Large-Scale Data Based On Modified Affinity Propagation Algorithm," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 6, no. 1, pp. 23-33, 2016, doi: doi:10.1515/jaiscr-2016-0003.
- [107] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 1177-1178.

- [108] F. Alharithi, A. Almulihi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Discriminative learning approach based on flexible mixture model for medical data categorization and recognition," *Sensors*, vol. 21, no. 7, p. 2450, 2021.
- [109] Y. Ao, H. Li, L. Zhu, S. Ali, and Z. Yang, "The linear random forest algorithm and its advantages in machine learning assisted logging regression modeling," *Journal of Petroleum Science and Engineering*, vol. 174, pp. 776-789, 2019.
- [110] S. Ray, "A quick review of machine learning algorithms," in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, 2019: IEEE, pp. 35-39.
- [111] Z. Xu, J. Xuan, J. Liu, and X. Cui, "MICHAC: Defect Prediction via Feature Selection Based on Maximal Information Coefficient with Hierarchical Agglomerative Clustering," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 14-18 March 2016 2016, vol. 1, pp. 370-381, doi: 10.1109/SANER.2016.34.
- [112] A. D. Hartanto, Y. Pristyanto, and A. Saputra, "Document Similarity Detection using Rabin-Karp and Cosine Similarity Algorithms," in *2021 International Conference on Computer Science and Engineering (IC2SE)*, 2021, vol. 1: IEEE, pp. 1-6.
- [113] IEEE, "IEEE Publication Services and Products Board Operations Manual 2021 " 2021. [Online]. Available: <https://www.ieee.org/content/dam/ieee-org/ieee/web/org/pubs/pspb/opsmanual.pdf>.
- [114] H. T. Y. Achsan, D. Kurniawan, D. G. Purnama, Q. K. D. Barcah, and Y. Y. Astoria, "Application of Natural Language Processing Using Cosine-Similarity Algorithm in Making Chatbot Information on the New Capital City of the Republic of Indonesia," in *2022 7th International Workshop on Big Data*

- and Information Security (IWBIS)*, 1-3 Oct. 2022 2022, pp. 1-6, doi: 10.1109/IWBIS56557.2022.9924902.
- [115] M. Alobed, A. M. Altrad, and Z. B. A. Bakar, "A Comparative Analysis of Euclidean, Jaccard and Cosine Similarity Measure and Arabic Wordnet for Automated Arabic Essay Scoring," in *2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP)*, 2021: IEEE, pp. 70-74.
- [116] L. Du and C. Hu, "Text similarity detection method of power customer service work order based on TFIDF algorithm," in *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2022: IEEE, pp. 978-982.
- [117] D. Guan, D. Liu, and W. Zhao, "Adversarial Detection based on Local Cosine Similarity," in *2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2022: IEEE, pp. 521-525.
- [118] T.-T. Ha, V.-N. Nguyen, K.-H. Nguyen, K.-A. Nguyen, and Q.-K. Than, "Utilizing sbert for finding similar questions in community question answering," in *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*, 2021: IEEE, pp. 1-6.
- [119] H. M. Oo and W. P. Pa, "Myanmar news retrieval in vector space model using cosine similarity measure," in *2020 IEEE Conference on Computer Applications (ICCA)*, 2020: IEEE, pp. 1-5.
- [120] S. J. Putra, M. N. Gunawan, and A. A. Hidayat, "Feature Engineering with Word2vec on Text Classification Using The K-Nearest Neighbor Algorithm," in *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, 2022: IEEE, pp. 1-6.

- [121] R. Akbar, T. N. Suharsono, and A. S. Indrapriyatna, "Essay Test Based E-Testing Using Cosine Similarity Vector Space Model," in *2022 International Symposium on Information Technology and Digital Innovation (ISITDI)*, 2022: IEEE, pp. 80-85.
- [122] H. Arabi and M. Akbari, "Improving plagiarism detection in text document using hybrid weighted similarity," *Expert Systems with Applications*, vol. 207, p. 118034, 2022.
- [123] F. Ullah, S. Jabbar, and L. Mostarda, "An intelligent decision support system for software plagiarism detection in academia," *International Journal of Intelligent Systems*, vol. 36, no. 6, pp. 2730-2752, 2021.
- [124] L. Ajallouda, K. Najmani, and A. Zellou, "Doc2Vec, SBERT, InferSent, and USE Which embedding technique for noun phrases?," in *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 2022: IEEE, pp. 1-5.
- [125] V. Ramnarain-Seetohul, V. Bassoo, and Y. Rosunally, "Work-in-Progress: Computing Sentence Similarity for Short Texts using Transformer models," in *2022 IEEE Global Engineering Education Conference (EDUCON)*, 2022: IEEE, pp. 1765-1768.
- [126] H. Veisi, M. Golchinpour, M. Salehi, and E. Gharavi, "Multi-level text document similarity estimation and its application for plagiarism detection," *Iran Journal of Computer Science*, vol. 5, no. 2, pp. 143-155, 2022.
- [127] C. Lupafya and D. Balasubramaniam, "A framework for considering uncertainty in software systems," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022: IEEE, pp. 1519-1524.

- [128] P. Schulam and S. Saria, "Can you trust this prediction? Auditing pointwise reliability after learning," in *The 22nd international conference on artificial intelligence and statistics*, 2019: PMLR, pp. 1022-1031.
- [129] S. Alwidian, M. Dhaouadi, and M. Famelis, "A vision towards a conceptual basis for the systematic treatment of uncertainty in goal modelling," in *Proceedings of the 12th System Analysis and Modelling Conference*, 2020, pp. 139-142.
- [130] M. d. Araujo, G. F. de Almeida, and J. L. Nunes, "Epistemology Goes AI: A Study Of GPT-3's Capacity To Generate Consistent and Coherent Ordered Sets of Propositions on Single-Input-Multiple-Outputs Basis," *Available at SSRN 4204178*, 2022.
- [131] M. Nursalman, J. Kusnendar, and U. F. Fadhila, "Implementation of k-nearest neighbor with cosine similarity for classification abstract international journal of computer science," in *2018 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2018: IEEE, pp. 43-48.
- [132] M. Merchant. "Semantic Similarity with BERT." https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/nlp/ipynb/semantic_similarity_with_bert.ipynb (accessed 2023-05-18, 2023).
- [133] Y. Lin *et al.*, "Blockchain-aided secure semantic communication for ai-generated content in metaverse," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 72-83, 2023.
- [134] T. Wu *et al.*, "A brief overview of ChatGPT: The history, status quo and potential future development," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 5, pp. 1122-1136, 2023.
- [135] POE.COM. "About." <https://poe.com/about> (accessed 2023-05-18, 2023).

- [136] F. M. Prentice and C. E. Kinden, "Paraphrasing tools, language translation tools and plagiarism: an exploratory study," *International Journal for Educational Integrity*, vol. 14, no. 1, pp. 1-16, 2018.
- [137] A. Poth, B. Meyer, P. Schlicht, and A. Riel, "Quality Assurance for Machine Learning – an approach to function and system safeguarding," in *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, 11-14 Dec. 2020 2020, pp. 22-29, doi: 10.1109/QRS51102.2020.00016.
- [138] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.
- [139] CityU, "Rules on Academic Honesty, City university of Hong Kong," 2020. [Online]. Available: https://www.cityu.edu.hk/provost/academic_honesty/rules_on_academic_honesty.htm.
- [140] Y. Qiu and B. Yang, "Research on micro-blog text presentation model based on word2vec and tf-idf," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2021: IEEE, pp. 47-51.
- [141] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via chatgpt," *arXiv preprint arXiv:2304.07590*, 2023.
- [142] A. Duque-Torres, D. Pfahl, C. Klammer, and S. Fischer, "Bug or not Bug? Analysing the Reasons Behind Metamorphic Relation Violations," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023: IEEE, pp. 905-912.
- [143] R. Alur *et al.*, "Syntax-guided synthesis," in *2013 Formal Methods in Computer-Aided Design*, 2013: IEEE, pp. 1-8.
- [144] S. Chaudhuri, K. Ellis, O. Polozov, R. Singh, A. Solar-Lezama, and Y. Yue, "Neurosymbolic programming," *Foundations and Trends® in Programming Languages*, vol. 7, no. 3, pp. 158-243, 2021.

- [145] L. Dong and M. Lapata, "Coarse-to-Fine Decoding for Neural Semantic Parsing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 731-742.
- [146] S. Gulwani, O. Polozov, and R. Singh, "Program synthesis," *Foundations and Trends® in Programming Languages*, vol. 4, no. 1-2, pp. 1-119, 2017.
- [147] P. Yin and G. Neubig, "A Syntactic Neural Model for General-Purpose Code Generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 440-450.
- [148] HuggingFace. "mrm8488/bert-small2bert-small-finetuned-cnn_daily_mail-summarization." HuggingFace. https://huggingface.co/mrm8488/bert-small2bert-small-finetuned-cnn_daily_mail-summarization (accessed 24/12/2023, 2023).
- [149] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [150] HuggingFace. "cahya/bert2gpt-indonesian-summarization." HuggingFace. <https://huggingface.co/cahya/bert2gpt-indonesian-summarization> (accessed 24/12/2023, 2023).
- [151] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.
- [152] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [153] HuggingFace. "AhmedSSoliman/MarianCG-CoNaLa-Large." HuggingFace. <https://huggingface.co/AhmedSSoliman/MarianCG-CoNaLa-Large> (accessed 24/12/2023, 2023).

- [154] A. S. Soliman, M. M. Hadhoud, and S. I. Shaheen, "MarianCG: a code generation transformer model inspired by machine translation," *Journal of Engineering and Applied Science*, vol. 69, no. 1, p. 104, 2022.
- [155] HuggingFace. "AhmedSSoliman/DistilBERT-Marian-Model-on-CoNaLa." HuggingFace. <https://huggingface.co/AhmedSSoliman/DistilBERT-Marian-Model-on-CoNaLa> (accessed 24/12/2023, 2023).
- [156] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
- [157] HuggingFace. "distilbert/distilbert-base-uncased." HuggingFace. <https://huggingface.co/distilbert/distilbert-base-uncased> (accessed 24/12/2023, 2023).
- [158] HuggingFace. "AhmedSSoliman/DistilRoBERTa-Marian-Model-on-CoNaLa." HuggingFace. <https://huggingface.co/AhmedSSoliman/DistilRoBERTa-Marian-Model-on-CoNaLa> (accessed 24/12/2023, 2023).
- [159] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," p. arXiv:1907.11692doi: 10.48550/arXiv.1907.11692.
- [160] HuggingFace. "distilbert/distilroberta-base." HuggingFace. <https://huggingface.co/distilbert/distilroberta-base> (accessed 24/12/2023, 2023).
- [161] HuggingFace. "AhmedSSoliman/ELECTRA-Marian-Model-on-CoNaLa." HuggingFace. <https://huggingface.co/AhmedSSoliman/ELECTRA-Marian-Model-on-CoNaLa> (accessed 24/12/2023, 2023).

- [162] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," *arXiv e-prints*, p. arXiv: 2003.10555, 2020.
- [163] HuggingFace. "google/electra-base-discriminator." HuggingFace. <https://huggingface.co/google/electra-base-discriminator> (accessed 24/12/2023, 2023).
- [164] HuggingFace. "AhmedSSoliman/LUKE-Marian-Model-on-CoNaLa." HuggingFace. <https://huggingface.co/AhmedSSoliman/LUKE-Marian-Model-on-CoNaLa> (accessed 24/12/2023, 2023).
- [165] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "LUKE: Deep contextualized entity representations with entity-aware self-attention," *arXiv preprint arXiv:2010.01057*, 2020.
- [166] HuggingFace. "studio-ousia/luke-base." HuggingFace. <https://huggingface.co/studio-ousia/luke-base> (accessed 24/12/2023, 2023).
- [167] HuggingFace. "google/pegasus-x-base." HuggingFace. <https://huggingface.co/google/pegasus-x-base> (accessed 24/12/2023, 2023).
- [168] J. Phang, Y. Zhao, and P. J. Liu, "Investigating Efficiently Extending Transformers for Long Input Summarization," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 3946-3961.
- [169] POE.COM. "Code-Llama-34b." <https://poe.com/Code-Llama-34b> (accessed 2024-01-15, 2024).
- [170] B. Roziere *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.
- [171] H. Touvron *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

- [172] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.
- [173] Microsoft. "Copilot." Microsoft.com. <https://www.microsoft.com/en-us/edge/features/copilot?form=MA13FJ> (accessed 19/08/2024, 2024).
- [174] Microsoft. "Overview of Microsoft Copilot." Microsoft.com. <https://learn.microsoft.com/en-us/copilot/overview?ns-enrollment-type=Collection&ns-enrollment-id=7rmiy0yrnw8d6> (accessed 19/08/2024, 2024).
- [175] J. Achiam *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [176] Microsoft. "Tackle any challenge with Copilot." Microsoft.com. <https://www.microsoft.com/en-us/microsoft-copilot/personal-ai-assistant> (accessed 19/08/2024, 2024).
- [177] J. Wang, Y. Peng, X. Le, C. Chen, and X. Guan, "Code Generation from Natural Language Using Two-Way Pre-Training," in *2023 15th International Conference on Advanced Computational Intelligence (ICACI)*, 2023: IEEE, pp. 1-6.
- [178] M. Evtikhiev, E. Bogomolov, Y. Sokolov, and T. Bryksin, "Out of the bleu: how should we assess quality of the code generation models?," *Journal of Systems and Software*, vol. 203, p. 111741, 2023.
- [179] M. Post, "A Call for Clarity in Reporting BLEU Scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 186-191.
- [180] GoogleCloud. "Evaluating models." Google. <https://cloud.google.com/translate/automl/docs/evaluate> (accessed 24/12/2023, 2023).

- [181] B. Chen *et al.*, "Codet: Code generation with generated tests," *arXiv preprint arXiv:2207.10397*, 2022.
- [182] B. Steenhoek, M. M. Rahman, R. Jiles, and W. Le, "An empirical study of deep learning models for vulnerability detection," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023: IEEE, pp. 2237-2248.
- [183] S. Tabassum, L. L. Minku, D. Feng, G. G. Cabral, and L. Song, "An investigation of cross-project learning in online just-in-time software defect prediction," in *Proceedings of the acm/ieee 42nd international conference on software engineering*, 2020, pp. 554-565.
- [184] H. Yamamoto, D. Wang, G. K. Rajbahadur, M. Kondo, Y. Kamei, and N. Ubayashi, "Towards privacy preserving cross project defect prediction with federated learning," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023: IEEE, pp. 485-496.
- [185] C. Shah, "From Prompt Engineering to Prompt Science With Human in the Loop," *arXiv preprint arXiv:2401.04122*, 2024.
- [186] F. Zhang, J. Liu, Y. Wan, X. Yu, X. Liu, and J. Keung, "Diverse title generation for Stack Overflow posts with multiple-sampling-enhanced transformer," *Journal of Systems and Software*, vol. 200, p. 111672, 2023.
- [187] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.
- [188] HuggingFace. "XLNet (base-sized model)." Hugging Face. <https://huggingface.co/xlnet/xlnet-base-cased> (accessed 2024).
- [189] A. Conneau *et al.*, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

- [190] HuggingFace. "XLM-RoBERTa (base-sized model)." Hugging Face. <https://huggingface.co/FacebookAI/xlm-roberta-base> (accessed 2024).
- [191] HuggingFace. "Google's T5 Version 1.1." Hugging Face. https://huggingface.co/google/t5-v1_1-base (accessed 2024).
- [192] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485-5551, 2020.
- [193] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.
- [194] HuggingFace. "OpenAI GPT 1." Hugging Face. <https://huggingface.co/openai-community/openai-gpt> (accessed 2024).
- [195] X. Li, Z. Li, H. Xie, and Q. Li, "Merging statistical feature via adaptive gate for improved text classification," in *Proceedings of the AAAI conference on artificial intelligence*, 2021, vol. 35, no. 15, pp. 13288-13296.
- [196] D. Gunning, "Machine common sense concept paper," *arXiv preprint arXiv:1810.07528*, 2018.
- [197] B. Li, Y. Hou, and W. Che, "Data augmentation approaches in natural language processing: A survey," *Ai Open*, vol. 3, pp. 71-90, 2022.
- [198] L. F. A. O. Pellicer, T. M. Ferreira, and A. H. R. Costa, "Data augmentation techniques in natural language processing," *Applied Soft Computing*, vol. 132, p. 109803, 2023.
- [199] J. Chen, D. Tam, C. Raffel, M. Bansal, and D. Yang, "An empirical survey of data augmentation for limited data learning in nlp," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 191-211, 2023.

- [200] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507-525, 2014.
- [201] A. R. Ibrahimzada, Y. Varli, D. Tekinoglu, and R. Jabbarvand, "Perfect is the enemy of test oracle," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 70-81.
- [202] HuggingFace. "BERT base model (uncased)." Hugging Face. <https://huggingface.co/google-bert/bert-base-uncased> (accessed 2024).
- [203] P. K. Dunn, "13.5 Identifying outliers," in *Scientific Research and Methodology: An introduction to quantitative research in science and health*, 2021.

Appendix A

Python commands for building Software Defect Prediction models in Chapter 3

Table A.1 shows the python commands with configurations used in the experiment of Chapter 3 to build the twenty Software Defect Prediction models.

Table A.1: List of python commands with configuration used

Machine learning algorithm	Abbreviation	Python Command with configuration
Gaussian Mixture Model with “diagonal” covariance matrix option	GMM(diag)	GaussianMixture(covariance_type='diag', random_state=42, n_init=10, max_iter=100, n_components=2)
Gaussian Mixture Model with “spherical” covariance matrix option	GMM(spherical)	GaussianMixture(covariance_type='spherical', random_state=42, n_init=10, max_iter=100, n_components=2)
Gaussian Mixture Model with “full” covariance matrix option	GMM(full)	GaussianMixture(covariance_type='full', random_state=42, n_init=10, max_iter=100, n_components=2)
Bayesian Gaussian Mixture Model with “diagonal” covariance matrix option	BayGM(diag)	BayesianGaussianMixture(covariance_type='diag', random_state=42, n_init=10, max_iter=100, weight_concentration_prior=1, n_components=2)
Bayesian Gaussian Mixture Model with “spherical” covariance matrix option	BayGM(spherical)	BayesianGaussianMixture(covariance_type='spherical', random_state=42, n_init=10, max_iter=100, weight_concentration_prior=1, n_components=2)
Bayesian Gaussian Mixture Model with “full” covariance matrix option	BayGM(full)	BayesianGaussianMixture(covariance_type='full', random_state=42, n_init=10, max_iter=100, weight_concentration_prior=1, n_components=2)

K-Means model using “lloyd” EM algorithm	Kmean(full)	KMeans(n_clusters=2, init="k-means++", n_init=10, max_iter=500, random_state=42, algorithm='full')
K-Means model using “elkan” EM algorithm	Kmean(elkan)	KMeans(n_clusters=2, init="k-means++", n_init=10, max_iter=500, random_state=42, algorithm='elkan')
Mini-Batch K-Means	MiniKM	MiniBatchKMeans(n_clusters=2, init="k-means++", n_init=10, max_iter=500, random_state=42)
Bisecting K-Means model using “lloyd” EM algorithm	BisectKM(full)	BKMeans(n_clusters=2, init="k-means++", n_init=10, max_iter=500, random_state=42, algorithm='full')
Bisecting K-Means model using “elkan” EM algorithm	BisectKM(elkan)	BKMeans(n_clusters=2, init="k-means++", n_init=10, max_iter=500, random_state=42, algorithm='elkan')
Spectral Clustering	Spectral	SpectralClustering(gamma= 10, n_clusters=2, affinity='rbf', assign_labels='discretize', n_jobs=-1, random_state=42)
Affinity Propagation Clustering	Affinity	AffinityPropagation(damping= 0.75, preference=-1e05, affinity = 'euclidean', random_state=42)
Agglomerative Clustering	Agglo	AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
Balanced iterative reducing and clustering using hierarchies	Birch	Birch(branching_factor = 50, n_clusters = 2, threshold=1.0)
Density-Based Spatial Clustering of Applications with Noise	DBscan	DBSCAN(algorithm="auto", n_jobs=-1, min_samples=5, eps=4)
Ordering Points to Identify the Clustering Structure	Optics	OPTICS(algorithm='auto', cluster_method='xi', metric='minkowski', n_jobs=-1, predecessor_correction=True, min_cluster_size=0.01, min_samples=5, eps=4)
MeanShift clustering	Meanshift	MeanShift(bandwidth=bdw, bin_seeding=True, n_jobs=-1)
Random Forests	RF	RandomForestClassifier(random_state=4487, n_estimators=100, max_depth=4, max_features=0.43411018169657967, min_samples_leaf=0.006923966890649835, min_samples_split=0.10768893766764986)
K-Nearest Neighbours	KNN	KNeighborsClassifier(n_neighbors=7)