*Empirical Software Engineering Report2:*

# Analyzing and Comparing Experiments

*181250060 蒋祚竑*
*181250066 李镔达(presenter)*
*181250090 刘育麟*
*181870108 李泳劭*

group: 15 year: 18

2021 年 5 月 28 日

# 1 Study Summary

## 1.1 overview

We collected totally 42 experimental studies papers published in 2018 while 6 of them from EASE, 19 from EMSE and 17 from ESEM.The full citations info list below.

表 1: ESEM

| ID | CITATION |
|---|---|
| 1 | Comparing techniques for aggregating interrelated replications in software engineering<br>Adrian Santos, Natalia Juristo<br>10.1145/3239235.3239239 |
| 2 | An empirical study of inadequate and adequate test suite reduction approaches<br>Carmen Coviello, Simone Romano, Giuseppe Scanniello<br>10.1145/3239235.3240497 |
| 3 | The effect of noise on software engineers' performance<br>Simone Romano, Giuseppe Scanniello, Davide Fucci, Natalia Juristo, Burak Turhan<br>10.1145/3239235.3240496 |
| 4 | A longitudinal cohort study on the retainment of test-driven development<br>Davide Fucci, Simone Romano, Maria Teresa Baldassarre, Danilo Caivano et al.<br>10.1145/3239235.3240502 |
| 5 | Applying pattern-driven maintenance: a method to prevent latent unhandled exceptions in web applications<br>Diogo S. Mendon?a, Tarcila G. da Silva, Daniel Ferreira de Oliveira et al.<br>10.1145/3239235.3268924 |
| 6 | Simultaneous measurement of program comprehension with fMRI and eye tracking: a case study<br>Norman Peitek, Janet Siegmund, Chris Parnin et al.<br>10.1145/3239235.3240495 |
| 7 | Improving problem identification via automated log clustering using dimensionality reduction<br>Carl Martin Rosenberg, Leon Moonen<br>10.1145/3239235.3239248 |

| | |
|---|---|
| 8 | Prediction of relatedness in stack overflow: deep learning vs. SVM: a reproducibility study<br>Bowen Xu, Amirreza Shirani, David Lo, Mohammad Amin Alipour<br>10.1145/3239235.3240503 |
| 9 | Are mutants really natural?: a study on how ”naturalness” helps mutant selection<br>Matthieu Jimenez, Thiery Titcheu Checkam, Maxime Cordy, Mike Papadakis et al.<br>10.1145/3239235.3240500 |
| 10 | Calibrating use case points using bayesian analysis<br>Kan Qi, Anandi Hira, Elaine Venson, Barry W. Boehm<br>10.1145/3239235.3239236 |
| 11 | No search allowed: what risk modeling notation to choose?<br>Katsiaryna Labunets<br>10.1145/3239235.3239247 |
| 12 | dentifying unmaintained projects in github<br>Jailton Coelho, Marco Tulio Valente, Luciana L. Silva, Emad Shihab<br>10.1145/3239235.3240501 |
| 13 | Assessing the effect of data transformations on test suite compilation<br>Panagiotis Stratis, Vanya Yaneva, Ajitha Rajan<br>10.1145/3239235.3240499 |
| 14 | Using experience sampling to link software repositories with emotions and work well-being<br>Miikka Kuutila, Mika V. Mäntylä, Maëlick Claes, Marko Elovainio, Bram Adams<br>10.1145/3239235.3239245 |
| 15 | Revisiting the size effect in software fault prediction models<br>Amjed Tahir, Kwabena E. Bennin, Stephen G. MacDonell, Stephen Marsland<br>10.1145/3239235.3239243 |
| 16 | utomatic topic classification of test cases using text mining at an Android smartphone vendor<br>Junji Shimagaki, Yasutaka Kamei, Naoyasu Ubayashi, Abram Hindle<br>10.1145/3239235.3268927 |

表 2: EASE

| ID | CITATION |
|---|---|

| | |
|---|---|
| 1 | Interrelating Use Cases and Associated Requirements by Links: An Eye Tracking Study on the Impact of Different Linking Variants on<br>the Reading Behavior<br>Oliver Karras, Alexandra Risch, Kurt Schneider<br>10.1145/3210459.3210460 |
| 2 | Using reasoning markers to select the more rigorous software practitioners' online content when searching for grey literature<br>Ashley Williams<br>10.1145/3210459.3210464 |
| 3 | Bug Localization with Semantic and Structural Features using Convolutional Neural Network and Cascade Forest<br>Yan Xiao, Jacky Keung, Qing Mi, Kwabena E. Bennin<br>10.1145/3210459.3210469 |
| 4 | An Inception Architecture-Based Model for Improving Code Readability Classification<br>Qing Mi, Jacky Keung, Yan Xiao, Solomon Mensah, Xiupei Mei<br>10.1145/3210459.3210473 |
| 5 | Usability Evaluation Framework for Mobile Apps using Code Analysis<br>Neeraj Mathur, Sai Anirudh Karre, Y. Raghu Reddy<br>10.1145/3210459.3210480 |
| 6 | Assessing the Effect of Device-Based Test Scheduling on Heterogeneous Test Suite Execution<br>Panagiotis Stratis, Gordon Brown<br>10.1145/3210459.3210481 |

表 3: EMSE

| ID | CITATION |
|---|---|
| 1 | A comprehensive study of pseudo-tested methods<br>Oscar Luis Vera-Perez, Benjamin Danglot, Martin Monperrus, Benoit Baudry<br>10.1007/s10664-018-9653-2 |
| 2 | An ensemble-based model for predicting agile software development effort<br>Onkar Malgonde, Kaushal Chari<br>10.1007/s10664-018-9647-0 |

3 Balancing the trade-off between accuracy and interpretability in software defect prediction
Toshiki Mori & Naoshi Uchihira
10.1007/s10664-018-9638-1

4 Detecting requirements defects with NLP patterns: an industrial experience in the railway domain
Alessio Ferrari, Gloria Gori, Benedetta Rosadini, Iacopo Trotta3 ·Stefano Bacherini, Alessandro Fantechi, Stefania Gnesi
10.1007/s10664-018-9596-7

5 Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments
Giuseppe Scanniello, Carmine Gravino, Marcela Genero3 · Jose A. Cruz-Lemus, Genoveffa Tortora, Michele Risi, Gabriella Dodero
10.1007/s10664-017-9591-4

6 Does syntax highlighting help programming novices?
Christoph Hannebauer, Marc Hesenius, Volker Gruhn
10.1007/s10664-017-9579-0

7 Early prediction of merged code changes to prioritize reviewing tasks
Yuanrui Fan, Xin Xia, David Lo, Shanping Li
10.1007/s10664-018-9602-0

8 Experimenting with information retrieval methods in the recovery of feature-code SPL traces
Tassio Vale, Eduardo Santana de Almeida
10.1007/s10664-018-9652-3

9 Using human error information for error prevention Wenhua Hu · Jeffrey C. Carver ·Vaibhav Anu ·Gursimran S.Walia · Gary L. Bradshaw
10.1007/s10664-018-9623-8

10 Improved representation and genetic operators for linear genetic programming for automated program repair
Vinicius Paulo L. Oliveira · Eduardo Faria de Souza ·Claire Le Goues ·Celso G. Camilo-Junior
10.1007/s10664-017-9562-9

11 On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files
Tomasz Kuchta · Thibaud Lutellier ·EdmundWong · Lin Tan · Cristian Cadar
10.1007/s10664-018-9600-2

12 Overfitting in semantics-based automated program repair
Xuan Bach D. Le · Ferdian Thung · David Lo ·Claire Le Goues
10.1007/s10664-017-9577-2

| | |
|---|---|
| 13 | Preventing duplicate bug reports by continuously querying bugreports<br>Abram Hindle · Curtis Onuczko<br>10.1007/s10664-018-9643-4 |
| 14 | Program comprehension of domain-specific and general-purpose languages: replication of a family of experiments using integrated development environments<br>Tomaz Kosar · Saso Gaberc · Jeffrey C. Carver ·Marjan Mernik<br>10.1007/s10664-017-9593-2 |
| 15 | Programmers do not favor lambda expressions for concurrent object oriented code<br>Sebastian Nielebock · Robert Heumuller ·Frank Ortmeier<br>10.1007/s10664-018-9622-9 |
| 16 | A controlled experiment on time pressure and confirmation bias infunctional software testing<br>Iflaah Salman · Burak Turhan · Sira Vegas<br>10.1007/s10664-018-9668-8 |
| 17 | Supporting the analyzability of architectural component models empirical findings and tool support<br>Srdjan Stevanetic · Uwe Zdun<br>10.1007/s10664-017-9583-4 |
| 18 | What can Android mobile app developers do about the energy consumption of machine learning?<br>Andrea McIntosh · Safwat Hassan · Abram Hindle<br>10.1007/s10664-018-9629-2 |
| 19 | ProMeTA: a taxonomy for program metamodels in program reverse engineering<br>Hironori Washizaki · Yann-Gaël Guéhéneuc · Foutse Khomh<br>10.1007/s10664-017-9592-3 |

## 1.2 our target papers

We choose 5 papers for further analysis and comparison.They are EASE[3], EMSE[3][8], ESEM[3][9].

**EASE[3]:Bug Localization with Semantic and Structural Features using Convolutional Neural Network and Cascade Forest**

**Abstract:**

Background: Correctly localizing buggy files for bug reports together with their semantic and structural information is a crucial task, which would essentially improve the accuracy of bug localization techniques.

Aims: To empirically evaluate and demonstrate the effects of both semantic and structural information in bug reports and source files on improving the performance of bug localization, we propose CNN Forest in- volving convolutional neural network and ensemble of random forests that have excellent performance in the tasks of seman- tic parsing and structural information extraction.

Method: We first employ convolutional neural network with multiple filters and an ensemble of random forests with multi-grained scanning to extract semantic and structural features from the word vectors derived from bug reports and source files. And a subsequent cascade forest (a cascade of ensembles of random forests) is used to further extract deeper features and observe the correlated relationships between bug reports and source files. CNN Forest is then empirically evaluated over 10,754 bug reports extracted from AspectJ, Eclipse UI, JDT, SWT, and Tomcat projects.

Results: The experiments empirical- ly demonstrate the significance of including semantic and structural information in bug localization, and further show that the proposed CNN Forest achieves higher Mean Average Precision and Mean Reciprocal Rank measures than the best results of the four current state-of-the-art approaches (NP- CNN, LR+WE, DNNLOC, and BugLocator).

Conclusion: CNN Forest is capable of defining the correlated relationships between bug reports and source files, and we empirically show that semantic and structural information in bug reports and source files are crucial in improving bug localization

**EMSE[3]:Balancing the trade-off between accuracy and interpretability in soft-ware defect prediction**

**Abstract:**

Background: Classification techniques of supervised machine learning have been successfully applied to various domains of practice. When building a predictive model, there are two important criteria: predictive accuracy and interpretability, which generally have a trade-off relationship. In particular, interpretability should be accorded greater emphasis in the domains where the incorporation of expert knowledge into a predictive model is required.

Aims: The aim of this research is to propose a new classification model, called superposed naive Bayes (SNB), which transforms a naive Bayes ensemble into a simple naive Bayes model by linear approximation.

Method: In order to evaluate the predictive accuracy and interpretability of the proposed method, we conducted a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets.

Results: A trade-off analysis between the accuracy and interpretability of different classification techniques was performed with a scatter plot comparing relative ranks of accuracy with those of interpretability. The experiment results show that the proposed method (SNB) can produce a balanced output that satisfies both accuracy and interpretability criteria.

Conclusions: SNB offers a comprehensible predictive model based on a simple and transparent model structure, which can provide an effective way for balancing the trade-off between accuracy and interpretability.

**EMSE[8]:Experimenting with information retrieval methods in the recovery of feature-code SPL traces**

**Abstract:**

Background: The information retrieval research provides alternatives to recover traces from existing software information. In the software product line (SPL) engineering context, further research on information retrieval methods is required to explore the existence of products’ source code and support the SPL adoption by providing traceability information.

Aims: This work evaluates the methods’ retrieval quality targeting the extraction of feature-code trace links in a set of variant-rich software development projects.

Method: We propose a research method comprising two experiments with five information retrieval methods (Classic vector, Latent semantic indexing, Neural network, Extended boolean and BM25), applied to forty-one projects. The SPLTrac suite automates the research operation, using the information retrieval methods results as input to perform a five-step quantitative data analysis procedure based on parametric and non-parametric statistical techniques. The quality measurement is expressed by four dependent variables: precision, recall, F-measure and execution time.Results With a homogeneous result for execution time between methods, there are discrepancies for the other metrics. While Extended boolean presents the best results for precision and F-measure, BM25 provides the greatest recall results.

Conclusions: Such evidence indicates there is not a dominant method in terms of retrieval quality, since they need further improvements to achieve better performance in all quality perspectives.

**ESEM[3]:The effect of noise on software engineers’ performance**

**Abstract:**

Background: Noise, defined as an unwanted sound, is one of the commonest factors that could affect people’s performance in their daily work activities. The software engineering research community has marginally investigated the effects of noise on software engineers’ performance.

Aims: We studied if noise affects software engineers’ performance in: *(i)* comprehending functional requirements and *(ii)* fixing faults in source code.

Method: We conducted two experiments with final-year undergraduate students in Computer Science. In the first experiment, we asked 55 students to comprehend functional requirements exposing them or not to noise, while in the second experiment 42 students were asked to fix faults in Java code.

Results: The participants in the second experiment, when exposed to noise, had significantly worse performance in fixing faults in source code. On the other hand, we did not observe any statistically significant difference in the first experiment.

Conclusions: Fixing faults in source code seems to be more vulnerable to noise than comprehending functional requirements.

**ESEM[8]:Are mutants really natural?: a study on how ”naturalness” helps mutant selection Abstract**

**Abstract:**

Background: Code is repetitive and predictable in a way that is similar to the natural language. This means that code is ”natural” and this ”naturalness” can be captured by natural language modelling techniques. Such models promise to capture the program semantics and identify source code parts that ‘smell’, i.e., they are strange, badly written and are generally error-prone (likely to be defective).

Aims: We investigate the use of natural language modelling techniques in mutation testing (a testing technique that uses artificial faults). We thus, seek to identify how well artificial faults simulate real ones and ultimately understand how natural the artificial faults can be. Our intuition is that natural mutants, i.e., mutants that are predictable (follow the implicit coding norms of developers), are semantically useful and generally valuable (to testers). We also expect that mutants located on unnatural code locations (which are generally linked with error-proneness) to be of higher value than those located on natural code locations.

Method: Based on this idea, we propose mutant selection strategies that rank mutants according to a) their naturalness (naturalness of the mutated code), b) the naturalness of their locations (naturalness of the original program statements) and c) their impact on the naturalness of the code that they apply to (naturalness differences between original and mutated statements). We empirically evaluate these issues on a benchmark set of 5 open-source projects, involving more than 100k mutants and 230 real faults. Based on the fault set we estimate the utility (i.e. capability to reveal faults) of mutants selected on the basis of their naturalness, and compare it against the utility of randomly selected mutants.

Results: Our analysis shows that there is no link between naturalness and the fault revelation utility of mutants. We also demonstrate that the naturalness-based mutant selection performs similar (slightly worse) to the random mutant selection.

Conclusions: Our findings are negative but we consider them interesting as they confute a strong intuition, i.e., fault revelation is independent of the mutants’ naturalness.

**Reasons:** There are 3 factors that we took into consideration when chose papers:

1.Clear experiment structure and description: It’s considerably difficult for us to read all papers word by word.So we prefer those papers that have a clear description helping us build our understanding about what they do, how they do and why they do towards their experiment easily.

2.Clear features and methodological characteristics: We prefer those papers whose methodological characteristics and features are easy to identify.To explain more concretely,we prefer those paper that use some features nouns as title of a section.

3.The difficulty to understand experiment context and process: Some experiments are hard to understand because we don’t have the pre-knowledge of the domain involved.So we prefer those experiments that are easier to understand.For example, EMSE[8]:Experimenting with information retrieval methods in the recovery of feature-code SPL traces and our software engineering III course assignment have the same study direction: information retrieval, and we use the same datasets.

### 1.3 Problems and gained knowledge:

There are 3 main problems:

1.There are large amount of papers need to be checked and read.The time cost is pretty high.

2.Some methods or theory used in the experiment are hard to understand.We don't master pre-knowledge to understand the experiment accurately.

3.Some experiments don't meet our standards.

Some research knowledge we gained:

1.When researchers want to test the performance of their method or model, they usually include other researchers' method/model as additional treaments to compare performance. And they may use multiple datasets to prevent the coincidence effect and show the great universality of their method/model.

2.When the experimental units are human, the theat to validity must be considered extensively and cautiously.For example, in ESEM[3]:The effect of noise on software engineers' performance, to avoid students' social or mental stress affect the experiment, researchers promise data gathering would be shared anonymously and in aggregated fashion, and this experiment won't effect students'scores.

# 2 Study Analysis and Comparison

## 2.1 Study Analysis

表 4: Methodological characteristics of EASE[3]

| Characteristics | EASE[3] |
|---|---|
| Experimental Units | use 5 open-source Java projects to evaluate the performance of CNN_Forest |
| Treatments | use CNN_Forest and other models which can extract features from bug reports and source files |
| Responses | The mean average precision (MAP) and mean reciprocal rank (MRR) are used to evaluate the performance of CNN_Forest |
| Context | make the before-fixed version of the source code in each project publicly available and provide the tool used to construct the dataset. adopt different strategies to extract features from bug reports and source files. compare MAP and MRR of different stradegies on the test set among these projects. |
| Design | Within-subject design |
| Balance | Each project uses the same data set and share the same treatments and criteria. When comparing, the same number of models/strategies is used on each one. |

表 5: Validity of EASE[3]

| Validities | EASE[3] |
|---|---|
| Conclusion Validity | |
| Internal Validity | The performance of our proposed model may be somewhat dependent on the performance of the word embedding techniques. We have checked the property of the adopted word embedding techniques before adopting them in our model. Second- ly, the hyper-parameters configuration set could introduce some bias in the experimental results. However, we set the parameters according to the suggestions of the existing stud- ies, which enabled our choices to be reasonable. |
| Construct Validity | When conducting experiments, we split the dataset in each project into validation set (oldest bug reports), training set(older bug reports) and testing set (newest bug reports) similar to the strategy used in the literature. However, the splitting percentage is different from that used in most machine learning techniques. ¡br¿In this paper, we used two measures (MAP and MRR) that are widely used in most bug localization studies |
| External Validity | They evaluated their model on five dataset from Java projects as many bug localization studies did and tried to report the general results. However, the selection of only five projects may have potentially limited representativeness. On the other hand, the performance of their model on other projects written in other programming languages is still un- known. They intend to examine our model on more projects, especially the ones written in, for example, C++, in a future study. |

表 6: Methodological characteristics of ESEM[3]

| Characteristics | ESEM[3] |
|---|---|
| Experimental Units | 2 groups of final-year undergraduate students in Computer Science, 55 students in experiment1 and 42 students in experiment2 |
| Treatments | Groups in experiment1 are asked to comprehend functional requirements, while in experiment2 groups are asked to fix faults in Java code. Group is exposed to noise while another control group is not in the same experiment. |
| Responses | |
| Context | Independently from the treatment, the participants in Exp1 performed comprehension task 1 (on M-Shop) in the first period, while comprehension task 2 (on Theater) in the second. The participants in Exp2 carried out bug fixing task 1 (on LaTazza) in the first period, while bug fixing task 2 (on AveCalc) in the second. |
| Design | Between-subject design |
| Balance | One group is exposed to noise while control group isn't |
| Orthogonality | Noise |

表 7: Validity of ESEM[3]

| Validities | ESEM[3] |
|---|---|
| Conclusion Validity | The implementation of a treatment(e.g., NOISE) might be not similar between different participants (i.e., reliability of treatment implementation). As shown in Section 3.7, we mitigate this kind of threat by implementing treatment/control as standard as possible over different participants. |
| Internal Validity | Social threats to validity might exist. The participants exposed to noise might be more motivated to accomplish the task (e.g., due to arousal effect) than the participants working in normal condition [6, 18]. On the other hand, the participants exposed to noise might give up performing as they would do under normal conditions (i.e., resentful demoralization). This kind of threat is present when the data analysis is conducted on the first period (i.e., when dealing with a statistically significant carryover). |
| Construct Validity | To deal with this kind of threat, we exploited metrics well known and adopted in the literature (e.g., [1, 30, 34]). As far as Exp2 is concerned, we considered only one metric to assess participants’ performances (i.e., mono-method bias). Although we did not inform the participants about our research goals, they were aware of being part of an investigation on noise effect. Thus, there could be the risk of hypotheses guessing.We informed students that achieved performance would not affect the SE course grades and gather data would be shared anonymously and in aggregated fashion (i.e., evaluation apprehension). |
| External Validity | The participants were sampled by convenience. Therefore, generalizing the results to a different population(e.g., professional software developers rather than students at the University of Basilicata) poses a threat of interaction of selection and treatment. Working with students also implies various advantages, such as: their homogeneous prior knowledge, the availability of a large number of participants [39] (55 and 42 participants in studies like ours is appreciable), and the possibility to test experimental design and initial hypotheses [36].The use of UML as modeling notation in Exp1 might threaten the generalizability of the results (i.e., interaction of setting and treatment). |

表 8: Methodological characteristics of ESEM[8]

| Characteristics | ESEM[8] |
|---|---|
| Experimental Units | 3 groups of different mutant selection strategies, additionally with 1 control group. All of them are based on a benchmark set of 5 open-source projects |
| Treatments | 3 groups conduct strategies that rank mutants according to a) their naturalness (naturalness of the mutated code), b) the naturalness of their locations (naturalness of the original program statements) and c) their impact on the naturalness of the code that they apply to (naturalness differences between original and mutated statements), and control group randomly selects mutants |
| Responses | |
| Context | Overall, to compute the naturalness of all mutants, we proceed as follows (for every mutated file): (1) Collect and tokenize all source code files of the projects containing the mutated file under evaluation (2) Exclude the file that has been mutated (3) Use the resulting set of source files as the training corpus to build two N-Gram models, one at the file-level and another one at the line-level(4) For the mutant files (test corpus): • Tokenize the mutant.• Compute its cross entropy as well as the original one using the file level model. • Do a diff between the line level tokenized versions of the original and the mutant. • Measure the cross entropy of the deleted lines and added lines using the line level model and attribute it to the original and mutant, respectively. |
| Design | Within-subject design |
| Balance | Repeatedly apply mutation testing by selecting the x% of the top rank mutants(we consider sets of 0, 5%, 10%, 15% to 100%) |
| Orthogonality | Strategies |

表 9: Validity of ESEM[8]

| Validities | ESEM[8] |
|---|---|
| Conclusion Validity | Regarding Kylm, the project is relatively old and not well documented but is still regularly used as comparison for new approaches like in the work of Pickhardt et al. [42], which shows that the results provided by the tool are still considered as relevant by the NLP community. To reduce this threat, we carefully analysed and tested the tool. Similarly, the use of 4-gram and of Modified Kneser Ney smoothing may have an impact on our results. We chose these options as suggested by our previous work [18]. |
| Internal Validity | A potential threat affecting the internal validity of our study stems from the sets of mutants and test suites that we used. We used state-of-the-art mutation testing tools [39] supporting all the mainstream mutation operators [23]. To compose the test pools, we used multiple test suites that were generated by state-of-the-art test generation tools, i.e., Randoop [36] and EvoSuite [13]. Although it is possible that different tests may lead to different results, the practice we employed reflects what current test case generation research has to offer in large-scale experiments. |
| Construct Validity | Threats that affect the construct validity of our study concerns the metrics we used. To evaluate mutant ranking we used fault-revealing mutants and fault revelation probabilities approximated by multiple test executions. We deem this metric appropriate since fault revelation forms the purpose of testing. |
| External Validity | The generalisability of the results is a common threat to the external validity of every experimental study. To mitigate this threat, we used real-world projects with real faults. |

表 10: Methodological characteristics of EMSE[8]

| Characteristics | EMSE[8] |
|---|---|
| Experimental Units | five information retrieval methods (Classic vector, Latent semantic indexing, Neural network, Extended boolean and BM25), applied to forty-one projects. |
| Treatments | SPLTrac, the tool which handles the instrumentation of this experiment. This is an open source tool suite that automates the operation of information retrieval algorithms, from data preparation to collection of results. |
| Responses | The quality measurement is expressed by four dependent variables: precision, recall, F-measure and execution time. |
| Context | In the software product line (SPL) engineering context, further research on information retrieval methods is required to explore the existence of products’ source code and support the SPL adoption by providing traceability information. |
| Design | Within-subject design |
| Balance | The selected experiment methods are: Classic vector (Salton et al. 1975), Latent semantic indexing (Furnas et al. 1988) and Neural network (Wilkinson and Hingston 1991) as algebraic methods; Extended boolean (Salton et al. 1983) as set theoretic method; and BM25 (Robertson et al. 2004) as probabilistic method. |

表 11: Validity of EMSE[8]

| Validities | EMSE[8] |
|---|---|
| Conclusion Validity | Conclusion validity issues involve the low power of the statistical analysis. The number of projects can be a threat to the hypotheses not rejected in this experiment. Aiming to address it, the easy extension of SPLTrac to incorporate new projects is being explored, and improved results will be reported in future work. In addition, we already provide a set of heterogeneous projects, with distinct characteristics in terms of languages, amount of code lines, numbers of features and variability representation technique. |
| Internal Validity | It provides a comprehensive representation on how the IR methods behave when applied tosuch different objects. The different statistical tests also provide strength to the results and conclusions of this study. |
| Construct Validity | Our assumption is that development projects in practice would hardly present any embodied and explicit variability information in the source code (e.g., a feature name explicitly declared to perform conditional compilation). Thus, aiming to mitigate bias in the experiment results, any explicit variability information in the file content is removed before analysis because they could impact on the methods results. For instance, source code lines with #ifdef followed by the feature name are removed because they will increase the counting number of feature occurrences for the methods and eventually improve their results. |
| External Validity | The interaction of selection and method is a key external validity issue, which concerns the representativeness of the population that must be generalized. However, the preprocessor projects represent real-world applications with thousands of lines of code and dozens or hundreds of features. Besides the differences, such antagonistic project profiles did not present differences in the statistical test results. |

表 12: Methodological characteristics of EMSE[3]

| Characteristics | EMSE[3] |
|---|---|
| Experimental Units | we conducted a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets. |
| Treatments | In this paper, we propose a new classification model that uses a two-step approach, where simple naive Bayes models (Section 3.1) or tree augmented naive Bayes models (Section 3.2) are firstly combined into an ensemble model (Section 3.3) and then transformed back into a simple naive Bayes model by linear approximation (Section 3.4). We focus on binary classification problems, because multiclass classification problems can be converted into binary problems by either one-versus-all (OVA), all-versus-all (AVA), error-correcting output-coding (ECOC), or generalized coding (Aly 2005). |
| Responses | The experiment results show that the proposed method (SNB) can produce a balanced output that satisfies both accuracy and interpretability criteria. |
| Context | Classification techniques of supervised machine learning have been successfully applied to various domains of practice. When building a predictive model, there are two important criteria: predictive accuracy and interpretability, which generally have a trade-off relationship. In particular, interpretability should be accorded greater emphasis in the domains where the incorporation of expert knowledge into a predictive model is required. |
| Design | Within-subject design |
| Balance | In our experiments, we use 13 real-world public datasets from two different data sources. One is the cleaned version of the NASA MDP datasets, created by Shepperd et al. (2013) and provided by Menzies et al. (2016), where the problem data (e.g., cases with either conflicting feature values or implausible values) and the useless data (e.g., the features with constant values and either identical or inconsistent cases) are removed. The other is the JIT datasets provided by Kamei et al. (2013), which contain various change-level metrics extracted from well-known open source projects. |

表 13: Validity of EMSE[3]

| Validities | EMSE[3] |
|---|---|
| Conclusion Validity | In order to evaluate the accuracy and interpretability of the proposed method, they conducted a comparative study using well-known classification techniques such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners, over 13 real-world public datasets. For the evaluation of predictive accuracy, they adopt a similar approach to Ghotra et al. (2015), which uses multiple repetition of k-fold cross-validation and a double Scott-Knott test. For the evaluation of interpretability, they use a qualitative approach that assessed the interpretability of different types of classification techniques based on Lipton (2016)’s categories, i.e., model transparency (simulatability), component transparency (decomposability), and algorithmic transparency. |
| Internal Validity | In order to strengthen the robustness and generalization of their results, researchers performed highly-controlled experiments by referring to Ghotra et al. (2015).The tuning of the configuration parameters of predictors may also influence our results. Their experiments basically followed the default parameter settings of WEKA’s implementation, but a more controlled tuning of the configuration parameters may be required. |
| Construct Validity | Researchers employed 13 well-known public datasets that have been used in various other studies. It is assumed that the use of well-known public datasets improves the reliability of the empirical results.We adopted the AUC as a performance measure. The AUC is one of the most commonly used metrics in software fault prediction studies (Malhotra 2015). It is known to perform well in practice and is often used when a general measure of predictability is desired (Fawcett 2006). Nevertheless, there are many other performance measures such as accuracy, sensitivity, specificity, precision, recall, G-mean, and F-measure, each of which has both strengths and shortcomings. Jiang et al. (2008a) suggest that performance measures should be selected based on the model evaluation criteria specific to each project.Comparing the interpretability of different classification techniques is a difficult task. |
| External Validity | Base on only 13 real-world public datasets for defect prediction studies in the software engineering domain. Although the datasets vary widely in size and distribution, this could be a potential threat to external validity.The selection of classification techniques could be another source of threat. To reduce it, our experiments included a wide variety of defect predictors, such as rule-based learners, decision trees, regression models, support vector machines, neural networks, Bayesian learners, and ensemble learners. |

## 2.2 Study Comparison

**1.Experiments units:**

There are 3 types of experiment units in our chosen studies:human, project.ESEM[3] is the only human-unit experiment. ESEM[8], EMSE[8] and EASE[3] are both code-unit experiments.They both used open-source projects to conduct experiment.And EMSE[3] use both NASA MDP datasets and JIT datasets.

**2.Treatments and design:**

ESEM[3] is the only between-subject-design experiment.It divides its units into 2 group.Each group must finish 2 tasks, while one task is under a noisy experiment and the other under normal experiment. In other words, in a specific task, one group was affected by noise while the other wasn't.So it's a typically between-subject design.Other 4 experiments are typically method/model performance experiment:use different methods/models to post multiple datasets and compare the results.So they are all within-subject design.

**3.External validity:**

we find that both human-unit experiment and code-unit experiments face the same external validity threat: if the sample and treat have a certain relationship that lead the experiment result to a certain direction,in other words, a threat of interaction of selection and treatment. And the only 2 way to solve this threat is to enlarge the units scales and repeat the experiment.

**4.Internal validity:**

Both human-unit experiment and code-unit experiment face the internal validity.Human-unit experiment have to consider the mentality and social actions.In ESEM[3], researchers take some actions:they set a long enough wash-out time between 2 tasks to neutralize carryover effect, monitor students and ban communication etc. Code-unit experiment have to consider whether some tools/techniques that used in data pre-prosess or other steps may influence the result.To avoid this problem, researchers repeat their experiments using different tools/techniques.For example, in ESEM[8], to compose test pools, researchers use multiple test suites generated by different test generation tools.