CS846-001 Week 9 Reviews

Jun Qing Lim
University of Waterloo
jq3lim@uwaterloo.ca

1 Who Should Fix This Bug?

1.1 Problem Being Solved

The paper presents a semi-automated approach to recommend potential candidates to fix a given bug. It aims to resolve the issue of lack of information in determining which developer is best suited to fix a bug, while taking up a significant time and cost in assigning a bug to a developer. This result in slower bug-fixing process and caused a large backlog of open and unresolved bugs. Although some research were being done in bug assignment, but little research were done on examining the cause of this inefficiency and most did not propose an actual solution.

1.2 New Idea Proposed

The authors identified from past records that experienced developers in solving bugs of a particular kind produce better high quality solutions in fixing bugs of the same kind and thus proposed a machine learning algorithm (using historical data) for such bug-fix assignment recommendations. It mainly consists of four steps:

1. characterizing bug reports to understand the pattern of similar bug reports and developers that solve them
2. assigning a label to each report to classify the report-developer assignment
3. choosing reports to train a (supervised) machine learning algorithm to refine the reports in preventing bias / ungeneralizable data
4. apply the algorithm to create classifier to recommend assignments using Support Vector Machine (SVM)

The authors then tested the accuracy of these assignments for Firefox and Eclipse bug repositories with more than 50% but not for gcc projects of only 6% accuracy due to skewed characteristics of gcc projects. It was hoped that the proposal will imprpove the efficiency and quality of bug fixing in software development.

1.3 Positive Points

Firstly, the paper did a thorough and detailed explanation in each decision made such as comparing SVM vs Naive Bayes vs C4.5 performance before deciding SVM to be the best algorithm, this gives a good elaboration on the proposed idea. Secondly, the paper used real-world open source repositories to conduct the study on the bug assignment algorithm, this makes the proposed algorithm to be relevant to the real software development world (instead of closed-software). Thirdly, the paper gave good critical discussions on the findings in assessing the value of the proposal, giving convincing reasons on the practical and actual effectiveness of the algorithm.

1.4 Negative Points

Firstly, the paper has a limited scope as it only examined / studied in three large open-source repositories thus may not be representative to all software development projects / domains. Secondly, the proposed algorithm only use historical data / reports to build the classifier and did not take in considerations of other factors such as difficulty of bug, competencies of developers, etc. whicy may affect the accuracy and reliability in producing optimal recommendations.

1.5 Future Work

One extension would be conduct on various domains of projects to have broader scope of recommendations. The other extension could be seeking survey or evaluation from developers to seek feedbacks on how accurate the proposed algorithm is, allowing us to improve its effectiveness.

1.6 Rating

5 out of 5. Well-written paper, although with common negative points as other papers, it was an idea which I have not thought of such recommendation possibility.

1.7 Discussion Points

- What are some real-world challenges with bug assignment in industry? Share the experiences.
- What other factors could affect how a bug could be efficiently fixed?
- With the rising developments in software, are there currently any similar solutions / tools practiced in industry?

2 Deep Learning based Vulnerability Detection: Are We There Yet?

2.1 Problem Being Solved

The paper presents a comprehensive survey on the effectiveness of deep learning (DL) based techniques in performing vulnerability prediction in real-world settings. As vulnerability detection is a critical task in software security, traditional approaches are often time-consuming and ineffective while DL-based techniques have shown promising results in improving accuracy of vulnerability detection, but there is still much work to be done to in developing robust detection performance. The authors aim to bridge this information gap by exploring multiple DL-based prediction techniques with its strengths and weaknesses to provide guidance for practitioners and researchers in DL-based prediction research.

2.2 New Idea Proposed

The authors first curated a comprehensive real world dataset (REVEAL) from open-source projects in building a model to detect the presence of real-world vulnerabilities in two phases:

1. **Feature extraction phase**: convert and extract features from code such as semantic, syntatic, granularity, etc to produce a graph
2. **Training phase**: using the extracted features to train a representation learning model to distinguish vulnerable and non-vulnerable code

The authors then experimented the model built, recorded the prediction results and obtained the following findings:

- neither pre-trained models or re-trained models of existing approaches used to predict vulnerabilities could accurately predict (with high generalization)
- these existing approaches of models pose limitations in detecting vulnerabilities because the models do not handle variations of data
- current approaches could be improved using the proposed model of REVEAL pipeline that handles data variations with improved accuracy up to 33% in precision and 128% in recall

Finally, the authors concluded that the accuracy of any prediction model built will depend on the use case scenario and the proposed REVEAL could act as a better vulnerability prediction tool as it is easily configurable.

2.3 Positive Points

Firstly, the authors not just proposed a new idea but also made its code and data open-source for the public, making it easy for researchers to conduct study in expanding the work. Secondly, the paper offered massive critical analysis such as elaboration on current deep-learning vulnerability detection approaches, its limitations and proposed a novel solution in addressing these challenges. Thirdly, the paper provided good guidance for researchers and practitioners working in the field of software security and laid out a good road-map for future work in DL-prediction.

2.4 Negative Points

Firstly, although paper's proposed idea was thoroughly discussed, it lacks a discussion on its future work such as how to expand from the authors' research work. Secondly, the length of the paper might be slightly overwhelming as it included many unnecessary details such s GPU formulas; as such it may not be easily comprehensible by all researchers.

2.5 Future Work

One extension is a research in integrating DL-based vulnerability detection mechanisms into current software security workflow and tools, giving a boost in productivity and efficiency across software teams. Furthermore, conduct a study to explore the strengths and weaknesses of DL-based vulnerability detection systems to provide better understanding of possible risks.

2.6 Rating

4 out of 5. Although slightly lengthy, but it is a clearly thought of and written paper with many interesting insights and usefulness.

2.7 Discussion Points

- Are there currently any similar DL-prediction practices in industry?
- Any ethical issues arise from using automated DL-based prediction mechanisms?
- What other factors could affect how vulnerable is a software?

3 Cross-project Defect Prediction

3.1 Problem Being Solved

The paper presents a large-scale study of cross-project defect prediction models to improve software quality and productivity. It conducts this study because there were limited studies being made on prediction models transferring from one project to another as most research has been focusing on within-domain predictions. Through this study, it hopes to address this limitation by exploring the effectiveness of cross-project defect prediction using machine learning and statistical tools to train for one project and predict defects in another project.

3.2 New Idea Proposed

The authors conducted the study and experiment on several open-source systems and commercial systems and built a defect prediction model based on logistics regression. Cross-project experiments were then conducted and a low accuracy or success rate was discovered. It then identifies that factors that affect whether cross-project predictions work such as: age of code base, code churn profiles, and code base characteristics. It also tries to improve the predictability by outlining a set of guidelines but little optimistic results were obtained. Finally, the authors concluded the following findings:

- same domain of project increases accuracy of cross-predictions
- precision of projects decrease when different domain projects are used
- success of cross-predictions are largely data driven
- there is no one single factor that lead to a successful cross-prediction

The authors hoped that its contribution on this empirical evaluation of cross-predictions could provide valuable insight to the software development field.

3.3 Positive Points

Firstly, the experiment was conducted on a large-scale dataset of different domains, this creates a good generalization information to other projects. Secondly, the paper proposed a novel approach of using transfer learning techniques to improve accuracy, as this is not a well-known technique thus made a remarkable contribution. Thirdly, although the paper's conclusion did not lead to a successful cross-prediction experiment, but it has outlined the possible extension work to this paper, making researchers an insight to improve the methods.

3.4 Negative Points

Firstly, the paper's elaboration on its methodology is little short such that it lacks analysis on the techniques or methods adopted, as such it may be difficult to understand the effectiveness of the adopted technique in the cross-project prediction. Secondly, the paper lacks thorough discussions over the results obtained from the experiment and did not address its research questions clearly, making it challenging to interpret the paper's contributions.

3.5 Future Work

One extension could be explore and investigate the effectiveness of transfer learning techniques (trained on one domain and predict on another domain), this allows us to better understand its practical applicability on a larger scale. Besides that, we could conduct an in-depth study to understand the factors that affect cross-project defect prediction as the paper lacks detail in identifying the factors, allowing us to develop methods in improving such predictions.

3.6 Rating

4 out of 5. The paper proposed a relevant discussion in defect predictions and highlighted important points in improving software quality (defect predictions) with convincing evidences.

3.7 Discussion Points

- Why do we need to have cross-project defect predictions?
- How widely is transfer learning technique (train on one domain and used in another domain) adopted in industry?
- Does cross-predictions always work?