

Security Bug Report Prediction Within and Across Projects: A Comparative Study of BERT and Random Forest

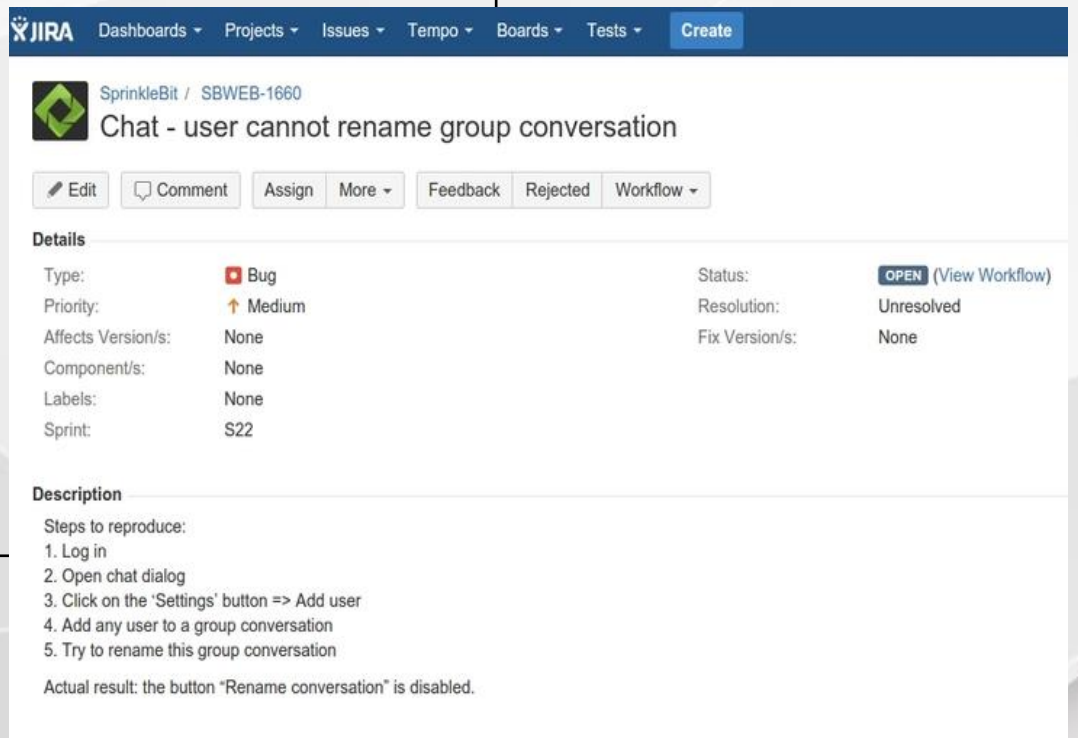
Farnaz Soltaniani [1], Mohammad Ghafari [2], Mohammed Sayagh [3]

[1 2] Technische Universität Clausthal, Institute for Software and Systems Engineering, Germany

[3] École de technologie supérieure - ÉTS Montréal, Canada

Problem statement

- Software systems constantly face problems and bug reports (BRs) are primary means to inform development teams of the defect
- Security bug reports (SBRs) are slow to resolve, leading to heightened vulnerability exposure
- Machine Learning techniques have been introduced to automatically classify SBRs from None SBRs (NSBRs)
- The scarcity of SBRs leads to a significant class imbalance



Where it All began?

Text Filtering and Ranking for Security Bug Report Prediction

Fayola Peters, Thein Than Tun, *Member, IEEE,* Yijun Yu, *Member, IEEE,* and Bashar Nuseibeh, *Member, IEEE*

Abstract—Security bug reports can describe security critical vulnerabilities in software products. Bug tracking systems may contain thousands of bug reports, where relatively few of them are security related. Therefore finding unlabelled security bugs among them can be challenging. To help security engineers identify these reports quickly and accurately, text-based prediction models have been proposed. These can often mislabel security bug reports due to a number of reasons such as class imbalance, where the ratio of non-security to security bug reports is very high. More critically, we have observed that the presence of security related keywords in both security and non-security bug reports can lead to the mislabelling of security bug reports. This paper proposes FARSEC, a framework for filtering and ranking bug reports for reducing the presence of security related keywords. Before building prediction models, our framework identifies and removes non-security bug reports with security related keywords. We demonstrate that FARSEC improves the performance of text-based prediction models for security bug reports in 90% of cases. Specifically, we evaluate it with 45,940 bug reports from Chromium and four Apache projects. With our framework, we mitigate the class imbalance issue and reduce the number of mislabelled security bug reports by 38%.

Index Terms—security cross words, security related keywords, security bug reports, text filtering, ranking, prediction models, transfer learning.

Data Quality Matters: A Case Study on Data Label Correctness for Security Bug Report Prediction

Xiaoxue Wu, Wei Zheng, Xin Xia, and David Lo

Abstract—In the research of mining software repositories, we need to label a large amount of data to construct a predictive model. The correctness of the labels will affect the performance of a model substantially. However, limited studies have been performed to investigate the impact of mislabeled instances on a predictive model. To bridge the gap, in this article, we perform a case study on the security bug report (SBR) prediction. We found five publicly available datasets for SBR prediction contains many mislabeled instances, which lead to the poor performance of SBR prediction models of recent studies (e.g., the work of Peters *et al.* and Shu *et al.*). Furthermore, it might mislead the research direction of SBR prediction. In this article, we first improve the label correctness of these five datasets by manually analyzing each bug report, and we find 749 SBRs, which are originally mislabeled as Non-SBRs (NSBRs). We then evaluate the impacts of datasets label correctness by comparing the performance of the classification models on both the noisy (i.e., before our correction) and the clean (i.e., after our correction) datasets. The results show that the cleaned datasets result in improvement in the performance of classification models. The performance of the approaches proposed by Peters *et al.* and Shu *et al.* on the clean datasets is much better than on the noisy datasets. Furthermore, with the clean datasets, the simple text classification models could significantly outperform the security keywords-matrix-based approaches applied by Peters *et al.* and Shu *et al.*

Index Terms—Security bug report prediction, data quality, label correctness

Simple text classification methods outperformed both the hyperparameter-tuned models and the data preprocessing techniques.

Methodology

Project	Description	BRs	SBRs	NSBRs	SBRs (%)
Chromium	Web browser called Chrome	41,940	808	41,132	1.92
Derby	A relational database management system	1,000	179	821	17.9
Camel	A rule-based routing and mediation engine	1,000	74	926	7.4
Ambari	Hadoop management web UI backed by its RESTful APIs	1,000	56	944	5.6
Wicket	Component-based web application framework for Java programming	1,000	47	953	4.7

Within Project Prediction (WPP)

Within Project

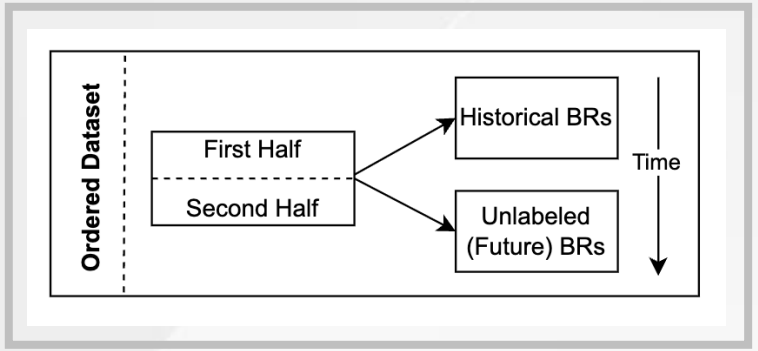
Prediction models are trained and tested using data from the same project.

SBR Augmentation

Training dataset are expanded with additional **SBRs** to assess how this increase affects prediction accuracy.

BR Augmentation

Training dataset are expanded with additional **SBRs and NSBRs** to assess how this increase affects prediction accuracy.



Cross Project Prediction (CPP)

Cross Project

Training dataset are expanded with additional **SBRs and NSBRs** to assess how this increase affects prediction accuracy.

Within Project Prediction (WPP)

BR Distribution in Within Project

Target Dataset	Train Set		Test Set	
	SBR (%)	NSBR (%)	SBR (%)	NSBR (%)
Chromium	371 (46%)	20599 (50%)	437 (54%)	20533 (50%)
Derby	82 (46%)	418 (51%)	97 (54%)	403 (49%)
Camel	28 (38%)	472 (51%)	46 (62%)	454 (49%)
Ambari	40 (72%)	460 (49%)	16 (28%)	484 (51%)
Wicket	24 (52%)	476 (50%)	23 (48%)	477 (50%)

10 possible models

BR Distribution using FARSEC

Target Dataset	Train Set		Test Set	
	SBR (%)	NSBR (%)	SBR (%)	NSBR (%)
Chromium	371 (45.9%)	20493 (49.9%)	437 (54.1%)	20533 (50.1%)
Derby	82 (45.8%)	46 (10.2%)	97 (54.2%)	403 (89.8%)
Camel	28 (37.8%)	214 (32%)	46 (62.2%)	454 (68%)
Ambari	40 (71.4%)	229 (32.1%)	16 (28.6%)	484 (67.9%)
Wicket	24 (51.1%)	207 (30.3%)	23 (48.9%)	477 (69.7%)

10 possible models

BR Distribution with Augmented SBRs and NSBRs

Target Dataset	Train Set		Test Set	
	SBR (%)	NSBR (%)	SBR (%)	NSBR (%)
Chromium	727 (62%)	24243 (54%)	437 (38%)	20533 (46%)
Derby	1067 (92%)	44373 (99%)	97 (8%)	403 (1%)
Camel	1118 (96%)	44322 (99%)	46 (4%)	454 (1%)
Ambari	1148 (99%)	44292 (99%)	16 (1%)	484 (1%)
Wicket	1141 (98%)	44299 (99%)	23 (2%)	477 (1%)

40+60+40+10=150 possible models

BR Distribution with Augmented SBRs

Target Dataset	Train Set		Test Set	
	SBR (%)	NSBR (%)	SBR (%)	NSBR (%)
Chromium	727 (62%)	20599 (50%)	437 (38%)	20533 (50%)
Derby	1067 (92%)	418 (51%)	97 (8%)	403 (49%)
Camel	1118 (96%)	472 (51%)	46 (4%)	454 (49%)
Ambari	1148 (99%)	460 (49%)	16 (1%)	484 (51%)
Wicket	1165 (98%)	476 (50%)	23 (2%)	477 (50%)

Adding both BRs yealds 150 models

Cross Project Prediction (CPP)

Bug Report Distribution for CPP Using All Datasets

Target Dataset	Train Set		Test Set	
	SBR (%)	NSBR (%)	SBR (%)	NSBR (%)
Chromium	356 (45%)	3644 (15%)	437 (55%)	20533 (85%)
Derby	985 (91%)	43955 (99%)	97 (9%)	403 (1%)
Camel	1090 (96%)	43850 (99%)	46 (4%)	454 (1%)
Ambari	1108 (99%)	43832 (99%)	16 (1%)	484 (1%)
Wicket	1117 (98%)	43823 (99%)	23 (2%)	477 (1%)

$$\binom{5}{1}\times 2+\binom{5}{2}\times 2+\binom{5}{3}\times 2+\binom{5}{4}\times 2=(10+20+20+10)=60$$

- CPP leverages labeled BRs from one or more projects to predict unlabeled future bug reports for a target project.

- We incrementally include additional BRs from other datasets during the training phase, utilizing their full training and testing splits. The latter half of one specific dataset is used exclusively for testing.

WPP Results

RQ1: How effectively can BERT predict SBRs compared to the state-of-the-art RF?

Dataset	Recall		Precision		F1-score		FPR		G-Measure	
	BERT	RF	BERT	RF	BERT	RF	BERT	RF	BERT	RF
Chromium	0.70	0.61	0.90	0.96	0.79	0.74	0.00	0.00	0.83	0.75
Derby	0.40	0.45	0.68	0.94	0.51	0.61	0.04	0.01	0.57	0.62
Camel	0.20	0.33	0.60	0.94	0.30	0.48	0.01	0.00	0.33	0.49
Ambari	0.25	0.44	0.31	0.37	0.28	0.40	0.02	0.02	0.40	0.60
Wicket	0.13	0.52	0.43	0.80	0.20	0.63	0.01	0.01	0.23	0.68
Average	**0.34**	**0.47**	**0.58**	**0.80**	**0.41**	**0.57**	**0.02**	**0.01**	**0.47**	**0.63**

BERT versus RF in WPP

- The results show that RF outperforms BERT with a 34% higher average G-measure for predicting SBRs in WPP.
- Effectiveness varies across different datasets.

- Applying FARSEC both BERT and RF exhibit improved average G-measure performance
- Except for the **Chromium** and **Derby** datasets, FARSEC improves the G-measure, differing from Wu et al.

O(N×W), where W represents the number of security-related keywords in the feature set, and N denotes the number of BRs in the training data

Dataset	Recall		Precision		F1-score		FPR		G-Measure	
	BERT	RF	BERT	RF	BERT	RF	BERT	RF	BERT	RF
Chromium	0.71	0.61	0.92	0.94	0.80	0.74	0.00	0.00	0.83	0.76
Derby	0.85	0.76	0.24	0.27	0.38	0.40	0.63	0.49	0.52	0.61
Camel	0.24	0.39	0.21	0.90	0.22	0.55	0.09	0.00	0.38	0.56
Ambari	0.50	0.69	0.10	0.44	0.16	0.54	0.16	0.03	0.63	0.81
Wicket	0.22	0.60	0.23	0.60	0.22	0.60	0.04	0.01	0.35	0.75
Average	**0.50**	**0.61**	**0.34**	**0.63**	**0.36**	**0.57**	**0.18**	**0.11**	**0.54**	**0.70**

BERT versus RF in WPP using FARSEC

Augmentation Results

RQ2: What is the impact of augmenting data on the performance of security bug report predictors?

Dataset	Recall		Precision		F1-score		FPR		G-Measure	
	BERT	RF	BERT	RF	BERT	RF	BERT	RF	BERT	RF
Chromium $_{SBRs}$	0.72	0.60	0.90	0.95	0.80	0.74	0.00	0.00	0.84	0.75
Chromium $_{BRs}$	0.73	0.62	0.88	0.95	0.80	0.75	0.00	0.00	0.84	0.77
Derby $_{SBRs}$	0.59	0.46	0.67	0.80	0.63	0.58	0.07	0.03	0.72	0.62
Derby $_{BRs}$	0.59	0.18	0.77	0.54	0.67	0.27	0.04	0.03	0.73	0.30
Camel $_{SBRs}$	0.28	0.45	0.37	0.60	0.32	0.51	0.05	0.03	0.43	0.61
Camel $_{BRs}$	0.41	0.26	0.73	0.92	0.53	0.40	0.01	0.00	0.58	0.41
Ambari $_{SBRs}$	0.31	0.68	0.11	0.23	0.17	0.35	0.08	0.07	0.46	0.78
Ambari $_{BRs}$	0.19	0.37	0.38	0.37	0.25	0.37	0.01	0.02	0.32	0.54
Wicket $_{SBRs}$	0.35	0.56	0.33	0.61	0.34	0.59	0.03	0.02	0.51	0.71
Wicket $_{BRs}$	0.70	0.17	0.84	0.67	0.76	0.28	0.01	0.00	0.82	0.29
Average $_{SBRs}$	**0.45**	**0.55**	**0.48**	**0.64**	**0.45**	**0.55**	**0.05**	**0.03**	**0.59**	**0.70**
Average $_{BRs}$	**0.52**	**0.32**	**0.72**	**0.69**	**0.60**	**0.41**	**0.02**	**0.01**	**0.66**	**0.46**

- Augmenting datasets with more SBRs **often** results in RF outperforming BERT in terms of G-measure across all datasets, except **Chromium** and **Derby**.
- Adding both SBRs and NSBRs, RF's average performance significantly drops to **0.46**, while BERT achieves its highest average performance of **0.66**, surpassing RF.
- Comparing **Augmentation** results with **WPP** results showed that:
  - Adding more **SBRs** improves the G-Measure for both **BERT** and **RF** in the Camel, Ambari, Derby, and Wicket datasets
  - Adding more **BRs** improves prediction performance for **BERT**, but decrease for **RF**
  - However in both cases the results in **Chromium** were almost unchanged.

CPP Results

RQ3: How do BERT and RF models compare for cross-project prediction?

Dataset	Recall		Precision		F1-score		FPR		G-Measure	
	BERT	RF	BERT	RF	BERT	RF	BERT	RF	BERT	RF
Chromium $_{\text{Derby}}$	0.27	0.32	0.06	0.22	0.10	0.26	0.09	0.02	0.42	0.48
Chromium $_{\text{Camel}}$	0.12	0.02	0.20	0.02	0.15	0.02	0.01	0.02	0.21	0.04
Chromium $_{\text{Ambari}}$	0.20	0.20	0.06	0.28	0.09	0.23	0.07	0.01	0.33	0.33
Chromium $_{\text{Wicket}}$	0.44	0.32	0.84	0.36	0.57	0.34	0.00	0.01	**0.61**	0.48
Derby $_{\text{Chromium}}$	0.07	0.03	0.44	0.75	0.12	0.05	0.02	0.00	0.13	0.06
Derby $_{\text{Camel}}$	0.20	0.35	0.68	1.00	0.30	0.51	0.02	0.00	0.33	**0.52**
Derby $_{\text{Ambari}}$	0.07	0.10	0.50	0.66	0.13	0.17	0.02	0.01	0.13	0.18
Derby $_{\text{Wicket}}$	0.32	0.45	0.79	0.72	0.46	0.55	0.02	0.04	0.48	0.61
Camel $_{\text{Chromium}}$	0.20	0.19	0.82	1.00	0.32	0.32	0.00	0.00	0.33	0.32
Camel $_{\text{Derby}}$	0.28	0.41	0.65	0.86	0.39	0.55	0.02	0.01	0.44	0.58
Camel $_{\text{Ambari}}$	0.07	0.04	0.30	0.50	0.11	0.08	0.02	0.00	0.13	0.08
Camel $_{\text{Wicket}}$	0.26	0.56	0.92	0.89	0.41	0.69	0.00	0.01	0.41	**0.72**
Ambari $_{\text{Chromium}}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ambari $_{\text{Derby}}$	0.06	0.62	0.12	0.43	0.08	0.51	0.01	0.03	0.11	**0.76**
Ambari $_{\text{Camel}}$	0.06	0.00	0.14	0.00	0.09	0.00	0.01	0.00	0.11	0.00
Ambari $_{\text{Wicket}}$	0.06	0.06	0.50	0.33	0.11	0.10	0.00	0.00	0.11	0.11
Wicket $_{\text{Chromium}}$	0.17	0.13	0.80	1.00	0.29	0.23	0.00	0.00	0.29	0.23
Wicket $_{\text{Derby}}$	0.57	0.52	0.57	1.00	0.57	0.68	0.02	0.00	**0.72**	0.68
Wicket $_{\text{Camel}}$	0.13	0.52	0.43	0.92	0.20	0.66	0.01	0.00	0.23	0.68
Wicket $_{\text{Ambari}}$	0.09	0.08	0.12	1.00	0.10	0.16	0.03	0.00	0.16	0.15
Average	**0.18**	**0.25**	**0.45**	**0.60**	**0.23**	**0.31**	**0.02**	**0.01**	**0.29**	**0.35**

- Building a model based on one project may not effectively predict SBRs in another.

Dataset	Recall		Precision		F1-score		FPR		G-Measure	
	BERT	RF	BERT	RF	BERT	RF	BERT	RF	BERT	RF
Chromium	0.68	0.46	0.32	0.30	0.43	0.36	0.03	0.02	0.80	0.63
Derby	0.44	0.15	0.90	0.54	0.59	0.24	0.01	0.03	0.61	0.27
Camel	0.43	0.28	0.83	0.93	0.57	0.43	0.01	0.00	0.60	0.44
Ambari	0.13	0.06	0.50	0.50	0.20	0.11	0.00	0.00	0.22	0.12
Wicket	0.74	0.30	0.57	0.70	0.22	0.60	0.04	0.01	0.84	0.47
Average	**0.48**	**0.25**	**0.62**	**0.59**	**0.49**	**0.31**	**0.02**	**0.01**	**0.62**	**0.38**

- Utilizing BRs from all other datasets for training, BERT outperforms RF's 0.38.
- For datasets with fewer SBRs, the top-performing model in the CPP surpassed the WPP results.

Conclusion

- RF outperformed BERT with a higher average G-measure for within-project predictions.
- Additional SBRs from various projects into the training data improved the performance of **both models.**
- Including more BRs significantly reduced **RF's** average performance, yet boosts **BERT** to its highest average performance.
- In cross-project SBR prediction, **BERT** also achieved a remarkable G-measure, substantially higher than **RF**.

The Paper

Farnaz Soltaniani

2022 - Present

Clausthal University of Technology, Clausthal, Germany