Gregor Engels, Regina Hebig, Matthias Tichy (Hrsg.): SE 2023,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 103

Variable Misuse Detection: Software Developers versus Neural Bug Detectors

Cedric Richter[1], Jan Haltermann[2], Marie-Christine Jakobs[3], Felix Pauck[4], Stefan Schott[5], Heike Wehrheim[6]

Abstract: Finding and fixing software bugs is a central part of software development. Developers are therefore often confronted with the task of identifying whether a code snippet contains a bug and where it is located. Recently, data-driven approaches have been employed to automate this process. These so called neural bug detectors are trained on millions of buggy and correct code snippets to *learn* the task of bug detection. This raises the question how the performance of neural bug detectors and software developers compare. As a first step, we study this question in the context of variable misuse bugs. To this end, we performed a study with over 100 software developers and two state-of-the-art approaches for neural bug detection. Our study shows that software developers are on average slightly better than neural bug detectors – even though the bug detectors are trained specifically for this task. In addition, we identified several bottlenecks in existing neural bug detectors which could be mitigated in the future to improve their bug detection performance.

Keywords: Bug detection; variable misuse bugs; empirical study

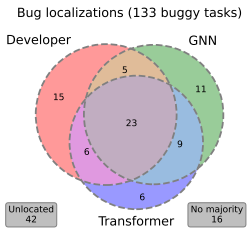
A Study of Developers and Neural Bug Detectors

Data-driven methods like neural bug detectors are becoming increasingly effective at the task of bug detection [He20]. Existing bug detectors often focus on frequent bug types such as variable misuses (a variable name is used although another was meant) or binary operator bugs (the wrong binary operator is used). While we can evaluate the bug detectors easily, e.g. on bugs mined from public repositories [KS20], it is unclear how human software developers would perform on these bug detection tasks and how they compare to existing neural bug detectors. To be able to compare software developers and neural bug detectors, we conducted a study with over 100 developers and evaluated them on the task of detecting variable misuse bugs in Java.

Our study was conducted in the form of a web survey, for which we developed a customized web interface. The interface is shown in Figure 1a. Here, the participant is shown a random

[1] University of Oldenburg, Oldenburg, Germany cedric.richter@uol.de
[2] University of Oldenburg, Oldenburg, Germany jan.haltermann@uol.de
[3] Technical University of Darmstadt, Darmstadt, Germany jakobs@cs.tu-darmstadt.de
[4] Paderborn University, Paderborn, Germany fpauck@mail.upb.de
[5] Paderborn University, Paderborn, Germany stefan.schott@upb.de
[6] University of Oldenburg, Oldenburg, Germany heike.wehrheim@uol.de

(a) User interface of the developer study



(b) Overlap of localized bugs

code snippet written in Java and must decide whether (1) the code snippet contains a variable misuse bug and (2) on which line the bug is located (if any). In total, participants are confronted with up to eight code snippets that were drawn randomly from a manually curated set of 310 possible code snippets and either contain a variable misuse bug or not. In the end, 304 snippets were solved by at least two participants. For each of these snippets, we computed the average developer performance (by aggregating all answers of this snippet) which we use to compare developers and bug detectors.

We compared the average developer performance with the performance of two state-of-the-art neural bug detectors on the same set of code snippets. Our results show that there is a significant overlap of bugs that can be detected by developers and bug detectors (e.g. as shown in Figure 1b). Still, we found that developers are generally better in avoiding false positives while maintaining a high bug detection rate. This observation has also led us to the discovery of several limitations of existing neural bug detectors such as (1) a misleading training distribution, (2) a missing robustness towards code length and (3) a lack of code context. A detailed evaluation and discussion of our results is available in [Ri22].

Data Availability

All our study results and an explorable version of the web survey are archived and available at Zenodo[7]. Our replication package also include participant answers (in an anonymous form).

Bibliography

[He20] Hellendoorn, Vincent J.; Sutton, Charles; Singh, Rishabh; Maniatis, Petros; Bieber, David: Global Relational Models of Source Code. In: ICLR. OpenReview.net, 2020.

[KS20] Karampatsis, Rafael-Michael; Sutton, Charles: How Often Do Single-Statement Bugs Occur?: The ManySStuBs4J Dataset. In: MSR. ACM, pp. 573–577, 2020.

[Ri22] Richter, Cedric; Haltermann, Jan; Marie-Christine, Jakobs; Felix, Pauck; Stefan, Schott; Wehrheim, Heike: Are Neural Bug Detectors Comparable to Software Developers on Variable Misuse bugs? In: ASE. 2022.

[7] https://doi.org/10.5281/zenodo.6958242