

Automating Precise Data Collection for Code Snippets with Bugs, Fixes, Locations and Types

Anonymous Authour(s)

I. BACKGROUND

The datasets for bugs and their fixes are essential for data-driven bug detection and auto-fix, which can be used for training pattern mining-based bug fix algorithms such as Getafix [3] and large-language-model based approaches [4]. Precisely locating the bugs and fixes can improve the precision of trained models, enhance the accuracy of bug detection and fix. Additionally, the bug types and commit messages reveal more concrete information of bugs, which could potentially benefit models for detection and fix, and enhance the efficiency of the debugging and code reviewing processes.

However, the granularity of current repository-mined bug-fixing datasets is usually at the function level [1], [2], without precise bug location and bug types. Thus, an approach for automatically and precisely mining code snippets with bugs, its fix, location and types from open-source repositories is still an open challenge.

II. CHALLENGE

The proposed challenge is automating precise data collection for code snippets with bugs, fixes, locations, bug types, and messages from open-source code repositories. It includes following specific requirements.

Automatic: The whole data collection process should be automated, with as less human-interaction as possible.

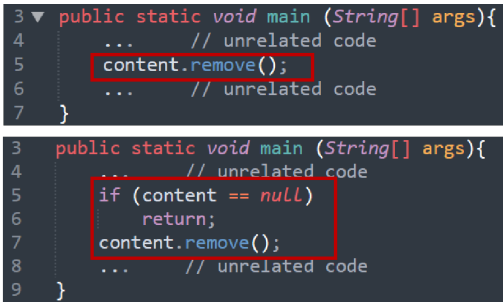
Precise: In the collected data, each data entry should include five elements (b, f, l, t, m), where:

- b is the *buggy code*, which is the bug code snippet with relative context.
- f is the *fixing code*, which is the bug fix snippet that correctly fixes the bug.
- l is the *location* of bug and fix in before code and after code.
- t is the *bug type* that might summarized from commits and merge-request(MR) messages or other potential sources.
- m (optional) is the related *message* inside the commits or MR message that describe the fix.

Language-agnostic: It is ideal that the solution is not designed for a specific programming language, but a language-agnostic method that fits for multiple programming languages. A downgrade requirement is that the method should at least be able to support one of the following programming languages: C/C++, Java, Python.

III. DELIVERABLES

The final deliverables should contain following two contents.

- 
- (1)

```
3 public static void main (String[] args){
4     ... // unrelated code
5     content.remove();
6     ... // unrelated code
7 }
```
- (2)

```
3 public static void main (String[] args){
4     ... // unrelated code
5     if (content == null)
6         return;
7     content.remove();
8     ... // unrelated code
9 }
```
- (3) Loc: Before: line 5; After: line [5, 8]
- (4) Type: Null pointer
- (5) Message: fix the null pointer in main function

Fig. 1. An example of one precise data entry. It includes (1) buggy codes, (2) fixing codes, (3) bug and fix locations, (4) bug types, and (5) related commit messages.

Methodology. A programming language-agnostic approach or algorithm that could automatically and precisely mining code snippets with bugs, fixes, locations types and messages from open-source repositories. It should include both the text instructions (a research paper would be preferred) and the tool implementation.

Dataset. A large dataset that include the precise code snippets with bugs, fixes, locations, bug types, and related messages that mined from open-source repositories using the proposed approach.

REFERENCES

- [1] Csuvik, V., and Vidács, L. (2022, May). FixJS: a dataset of bug-fixing JavaScript commits. In Proceedings of the 19th International Conference on Mining Software Repositories (pp. 712-716).
- [2] Tufano, M., Pantiuchina, J., Watson, C., Bavota, G., and Poshyvanyk, D. (2019, May). On learning meaningful code changes via neural machine translation. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 25-36). IEEE.
- [3] Bader, J., Scott, A., Pradel, M., and Chandra, S. (2019). Getafix: Learning to fix bugs automatically. Proceedings of the ACM on Programming Languages, 3(OOPSLA), 1-27.
- [4] Dawn Drain, Chen Wu, Alexey Svyatkovskiy, and Neel Sundaresan. 2021. Generating Bug-Fixes using Pretrained Transformers. In Proceedings of the 5th ACM SIGPLAN International Symposium on Machine Programming (MAPS '21), June 21, 2021, Virtual, Canada. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3460945.3464951>