Retrieving Self-executable and Functionally Correct Code to Improve Source Code Search

Abdus Satter, M G Muntaqeem, Nadia Nahar and Kazi Sakib
Institute of Information Technology, University of Dhaka

Abstract

Developers need to put lots of time and effort to reuse the code snippets retrieved by the existing code search engines. The reason is that these engines do not provide self-executable, functionally correct and easily understandable code snippets as search results. Developers manually resolve all the dependencies to make the code snippets executable in their development contexts. They have to write and execute the same test cases many times to check the correctness of the code fragments. In this paper, a technique has been proposed that converts each method in a code base into self-executable method (i.e., program slice) by resolving method calls, data and library dependencies. To ensure that the methods are functionally correct, automatic test scripts are generated and executed for each self-executable method based on the branch, statement, and path coverage. The understandability of the code fragments is increased by replacing irrelevant textual keywords with relevant words. All the self-executable code fragments are indexed using traditional Information Retrieval approach. So, when a user query is submitted, the technique will retrieve self-executable and functionally correct code snippets.

Introduction

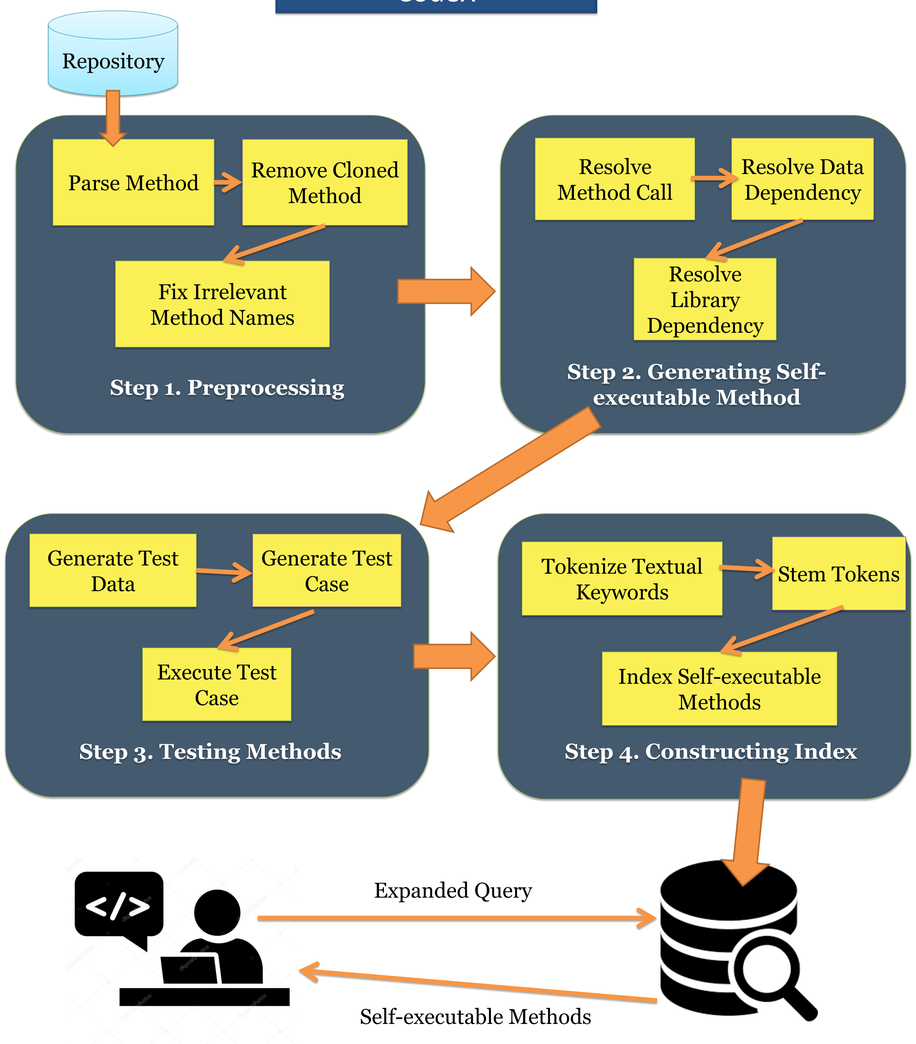
Developers search for code fragments to understand the usage of an API, to directly reuse in their development context, or to know the implementation of a particular functionality. Existing code search engines employ Information Retrieval (IR) approach to index source codes and query over the index. In these approaches, code snippets are considered as plain text document. When a query is submitted, these engines locate the relevant code fragments in a repository or retrieve these code snippets as search results. Developers spend significant amount of time and effort to make the retrieved code fragments executable in their development contexts through inspecting dependencies in the source files. Again, retrieved code fragments may have bugs that are not checked by the code search engines. Developers need to write and execute test cases to ensure the retrieved code fragments meet their needs. Many developers search for the same code fragment multiple times, and almost the same test cases are executed many times which costs lots of time, effort and resource.

Code search engines should retrieve functionally correct, easily comprehensible, and self-executable code fragments so that developers can easily reuse with minimum time and effort. However, several challenges are associated with that. The first challenge is to make the retrieved code fragments self-executable. The second challenge is to provide tested code fragments so that developers do not need to check by running the same test cases many times. The third challenge is to generate test cases automatically for checking the correctness of the code fragments. The fourth challenge is to fix the names of the code entities that have been written using irrelevant naming conventions.

Problem

- Testing retrieved code snippets in the development context is time consuming
- Inappropriate names of the code entities make these difficult to comprehend
- Same test cases are required to be run to check the correctness of the retrieved code snippets
- Many relevant code snippets cannot be retrieved due to having irrelevant textual keywords

Proposed Technique: CodeX



Implementation Details

- Clone Method Detection – SourcereCC [1]
- Program Slicing – Extract Method Refactoring Technique [2]
- Test Case Generation - Evosuite [3]
- Fixing Irrelevant Names - Accurate Method Name Suggestion Technique [4]

Conclusion

This paper presents a technique that improves the existing code search engines by converting the methods in a code base into the self-executable code fragments. Next, it checks the correctness of the code fragments through generating and executing test cases. As a result, the technique will retrieve self-executable and tested codes against the search query. Although the technique seems to be computationally expensive, it needs to be executed one time for a method during index construction. Currently, the technique is being implemented as an eclipse plugin. In future, it will be evaluated on open source and industrial projects to assess its effectiveness.

References

[1] Hitesh Sajnani, Vaibhav Saini, Jeffrey Svajlenko, Chanchal K Roy, and Cristina V Lopes. Sourcerercc: Scaling code clone detection to big-code. In Proceedings of the 38th International Conference on Software Engineering, pages 1157–1168. IEEE, 2016.
[2] Nikolaos Tsantalis and Alexander Chatzigeorgiou. Identification of extract method refactoring opportunities for the decomposition of methods. Journal of Systems and Software, 84(10):1757–1782, 2011.
[3] Gordon Fraser and Andrea Arcuri. Evosuite: automatic test suite generation for object-oriented software. In Proceedings of the 19th ACMSIGSOFT symposium and the 13th European conference on Foundations of software engineering, pages 416–419. ACM, 2011.
[4] Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. Suggesting accurate method and class names. In Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, pages 38–49. ACM, 2015

Presenter

Abdus Satter
bit0401@iit.du.ac.bd
Institute of Information Technology
University of Dhaka
Dhaka 1000, Bangladesh