Code improvement report

Roblox prime numbers:

Oleh Basystyi
Anna Stasyshyn
Artur Rudish
Anton Valihurskyi
Maksym Zhuk

April 2024

+

1 Introduction

In this little research, our team researched AI's capability to write unittests. In this part, we tested only **ChatGPT-3.5** Here, we will provide only aggregate conclusions about *test creation, test diagnosis & fix, coverage and optimization*. Full list of used tasks with some statistical data as *used number of prompts, number of test cases*, and a detailed summary of each task can be found in table on the third sheet. Also, we have the GitHub where you can find used problems code, tests and AI's optimization of existing tests.

2 Test creation

Generally, ChatGPT demonstrated a decent capability of test writing. It can properly identify ordinary test cases (as object initialization, special methods tests or simple function calls) as well as corner cases tests. For example, in problem Line ChatGPT created by itself for edge cases tests like intersection of collinear or perpendicular lines. Moreover, AI suggests writing performance test cases. But, from the first prompt, you can't get all the test cases that ChatGPT can give you. To extend test case base, you should better prompt the following:

- Do you have any suggestions to make more comprehensive tests?
- Can you add these cases to existing?
- Ensure that corner cases for {method} are satisfied
- Add test for {edge case}

These two prompts combined worked far better than simple "Make test cases more comprehensive.". We assume it's because ChatGPT was trained more on articles, books forums discussions with code examples, so it is easier for it to firstly make linkage to the description of required tests and only then write code by that description.

3 Test diagnosis & fix

Sometimes (actually usually) ChatGPT creates invalid tests. In most cases, it happens when you ask AI to extend test cases. We assume that's because AI can't process all consequences of its decisions. Alas, tested LLM can't resolve the problem even if user points to the invalid test case. If you write something like: "Test {test_name}" is invalid, ChatGPT will send you the same incorrect code, even if you said again that test is wrong. So, when fixing unittests with AI. You should try the following prompts:

- Your test {function_name} works incorrectly, because {reason}
- In test {function_name} you do {this}, but should do {that}
- You have an error on line {line number} {error exception}

4 Coverage

ChatGPT showed a great capability of writing tests with 100% coverage. Even if it didn't manage to cover all code from one prompt, another pair of messages, is needed to cover 100% of code. Although ChatGPT can cover 100% of code, it correctly identify what code is not covered in other cases. We explain it this way. When GPT sees a problem to which it must write tests, it doesn't really go through code, analyses connections as humans do, It rather gets the code, and "search" in train data, similar cases, so 100% coverage is not a result of code analysis, but rather mixing of AI knowledge base to particular task. Also, that's why, prompt like "your test cases do not cover line 42 doesn't work", user also must provide a reason why code on 42nd line doesn't work

The following prompts can be helpful, when someone tries to increase code coverage with ChatGPT:

- The same prompts as in **Test creation** section
- Your tests has small coverage, fix
- Your tests doesn't cover line [**line number**], because [**reason**]

5 Optimization

In most cases, ChatGPT showed no capabilities of test optimizations. It could only make style improvement, or adding setUp() and tearDown() methods. Of course, when user asks AI to optimize code, it returns "optimized" version with lots of invalid test cases, that again must be fixed manually. Only in one problem Equation (see in table on the third sheet), AI decided to split huge TestCase into smaller, and it somehow decreased runtime from 0.7 to 0.6 seconds. But, in general, we see no benefit of optimizing unittests with ChatGPT.

6 Recommendations & Conclusions

To sum up, AI is a powerful tool for generating simple unittest. Despite that fact, it has many disadvantages: can't really analyze code and usually sends invalid test cases that can be fixed only with manual directions. Thus, we conclude that AI should be used to generate unittest templates (to avoid routine work of writing test for special methods) that further must be reviewed and finalized by QA.