LOG6305 - Techniques avancées de test du logiciel

Assignment 5

Automatic test generation

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Winter 2024

1 Introduction

In this assignment, you will use all your skills developed through labs 2-4 to build a test generator. You will be tasked to combine fuzzing or search-based whole test suite generation (or both) to improve the output produced by a recommended Large Language Model (LLMs).

2 Objectives

The objectives of this lab are :

1. Explore the current stat-of-the-art LLMs, such as Google Gemini [1].
2. Apply the knowledge of advanced testing techniques to a practical use case.

3 The tasks

3.1 Required set-up

Operational system : Linux/macOS/Windows, **Python** : version = 3.9. Clone the following repository : `https://github.com/log6305/HIV_2024_TP5`

Setting Up the LLM Navigate to Google Gemini LLM documentation `https://ai.google.dev/pricing` and follow the instructions to set up the API key and launch a basic example `https://ai.google.dev/tutorials/quickstart`. In this lab, it is mandatory to use the free version of Gemini 1.0 Pro model. It has a restriction of 15 requests per minute and 1500 total requests per day.

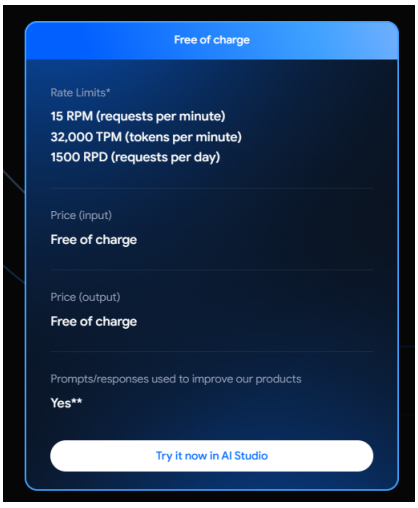

FIGURE 1 – Gemini 1.0 Pro model to use in the lab (free of charge version)

1. `https://ai.google.dev/pricing`

3.2 Your tasks

1. Question 1 (Exploration) : Clone the repository and run the 'llm_example.py' file. This example demonstrates how to use the given LLM to generate tests and how to evaluate the coverage achieved by the tests (1 point).
2. Question 2. In the 'to_test' folder two functions are given : number_to_words.py and strong_password_checker.py. Your goal is to test these functions. A baseline code is provided in 'generate_tests.py' file that generates a tests for a given function with LLM. The function is saved in the specified file ('test_generated.py'). Your task is to fill the TODO section of the file. You need to develop your test generator that will parse the tests produced by the LLM and use them to perform a search in order to improve the initial line as well as the branch coverage achieved by the tests. Moreover, you should minimize the number of test inputs required to achieve a certain level of coverage. Your generator should use either fuzzing or a search-based technique. At the output, your test generator should provide a list of inputs that can be used to test the function. The given inputs should achieve a better line and branch coverage, than the initial tests generated by the LLM. Based on your inputs a score will be calculated and displayed in the terminal. In your generator, you should use the "AbstractExecutor" object provided :

   ```
   # define the executor to be used with your test generator
   executor = AbstractExecutor(function_to_test)
   ```

   It has a limit of 100 evaluations. Once you evaluate the coverage of your tests for more, than 100 times, test generation will be stopped. You should take it into account and stop your generator before the budget is exceeded. You should run your test generator 5 times for each program under test and report the best obtained value. In your report include boxplots that show the score values obtained in the 5 runs (10 points).
3. Question 3. Describe the principle of operation of your generator in the report. We recommend to write its pseudocode (4 points).
4. Question 4. You will be given additional points based on the performance of your test generators on the two functions under test (average of the best score obtained for each function). Your additional points will be proportional to the ranking. The maximum additional points you can achieve equal to 5% of the assignment grade.

Additional instructions :

1. If needed, you can modify the input prompt to the LLM. But you should use only one prompt.
2. You can use either fuzzing or search based (or combination of both) for your test generation.

4 Expected deliverables

The following deliverables are expected :

— Link to your repository with the solution to Q2 or a .zip archive of the repository.
— The report for this practical work in the PDF format.

You should submit all the files to Moodle. When adding the PDF file, do not put into a zip. Additionally, you should add a ".txt" file with the following content : "Your full name, your student id" (this might be used for the grading automation). This assignment is individual.

The report file must contain the title and number of the laboratory, your name and student id.

5 Important information

1. Check the course Moodle site for the file submission deadline
2. A delay of [0.24 hours] will be penalized by 10%, [24 hours, 48 hours] by 20% and more than 48 hours by 50%.
3. No plagiarism is tolerated (including ChatGPT). You should only submit code created by you.