LLM-Based Code Generation Method for Golang Compiler Testing

Qiuhan Gu
State Key Laboratory for Novel Software Technology, Nanjing University, China
qiuhan.gu@smail.nju.edu.cn
Advisor: Yu Wang

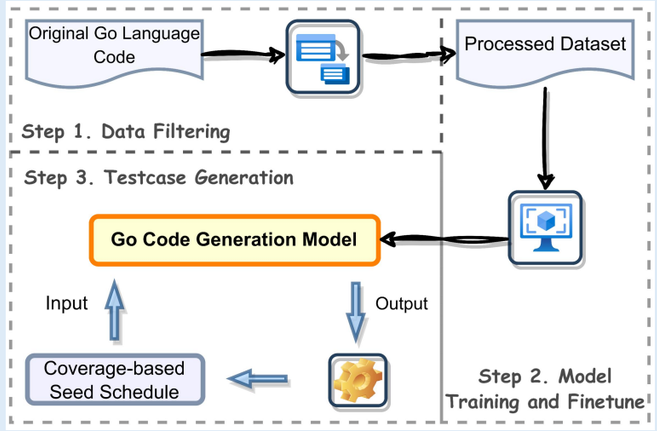
Association for Computing Machinery

Motivation

- Limited coverage and quantity of testcases generated by traditional testing method.
- Undefined behavior and syntax errors[1] in generated testcases.

Objective

A LLM-based high-quality code generation method.



Workflow of our method:

- **Step1:** Remove undefined behaviors and syntax errors from the dataset;
- **Step2:** Finetune a pre-trained model by the processed dataset;
- **Step3:** Generate testcases with a loop using coverage-based seed schedule.

Data Filtering

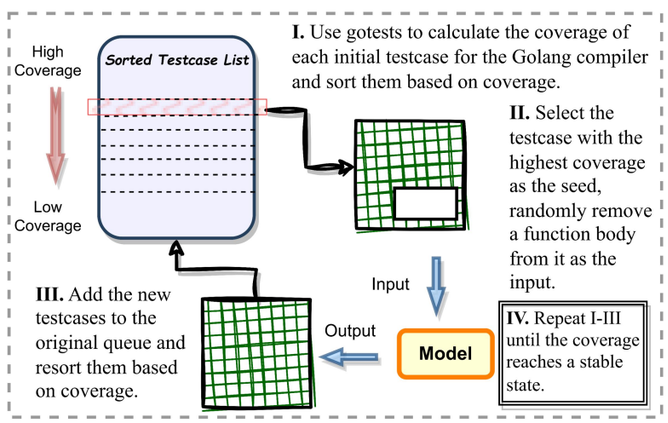
We designed a series of filtering criteria to remove low-quality programs:

- Remove files with syntax errors.
- Remove files with a character length exceeding 10000.
- Remove files with duplicate code[3].
- Remove files with an alphanumeric characters ratio below 0.25.
- Remove files that may contain undefined behavior.
- Remove files that reference the "internal" package.

Model Training and Finetuning

In our experiment, we finetuned the CodeT5[2] through 30 epochs, at the learning rate of 10−5 and warmup steps of 1000. Then we implemented our method on the Golang compiler, looping 1000 times to generate 1157 testcases (cost about 100h).

Coverage-based Seed Schedule



"Coverage" here refers to the measure of how much of the source code of the Golang compiler is exercised by the given testcases.

Performance

We employed the method on the Golang compiler, producing testcases that achieved an average coverage of 3.38%, compared to testcases with 0.44% average coverage generated by Go Fuzzing[4]. Among these testcases, only 2.79% exhibited syntax errors, and none manifested undefined behavior.

Our Web: https://github.com/GuQiuhan/LLM-Based-Code-Generation-Method-for_x0002_Golang-Compiler-Testing.

Future Work

- Correct model bias with attention mechanisms.
- Optimize the seed schedule.

Reference

[1] 2023. Go. https://go.dev/ref/spec.
[2] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. arXiv preprint arXiv:2109.00859 (2021).
[3] Miltiadis Allamanis. 2019. The Adverse Effects of Code Duplication in Machine Learning Models of Code. arXiv:1812.06469 [cs.SE]
[4] 2023. GoFuzzUrl. https://github.com/dvyukov/go-fuzz.