Automated Test Generation for Open-Source Projects using Natural Language Processing

Alexander Barajas-Ritchie
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, United States
barajale@oregonstate.edu

***Abstract*—**
***Index Terms*—**

I. Introduction

A. Problem Statement

How can natural language processing (NLP) be harnessed to automate test generation and enhance test coverage from comments and docstrings in open-source software projects? This initiative aims to bridge the gap between informal software documentation and formal testing procedures, enabling the generation of targeted tests that improve coverage and error detection. Specifically, the project will focus on:

- **Test Generation:** Creating tests that verify the consistency of code behavior with its described functionality in comments and docstrings.
- **Test Enhancement:** Augmenting existing tests to include checks for common coding issues such as boundary conditions, null pointer exceptions, and concurrency problems, which are frequently under-tested in open-source environments.

This approach seeks to leverage helpful yet underutilized natural language descriptions within code, enhancing the reliability and maintainability of open-source software.

B. Motivation

The development of open-source software plays a critical role in technological innovation and academic research. However, these projects often suffer from inadequate test coverage, which can lead to persistent bugs and security vulnerabilities. This project addresses these challenges by proposing a tool that leverages natural language processing to automatically generate and enhance test cases based on the existing comments and docstrings within the codebase.

The significance of this research lies in its potential to:

- **Reduce Technical Debt:** Well-structured test cases can significantly reduce ”technical debt,” a metaphor reflecting the eventual cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer.
- **Improve Code Quality:** By enhancing test coverage, the tool can help detect and eliminate ”code smells”—symptoms in the code that may indicate deeper problems.
- **Facilitate Continuous Development:** Effective testing ensures that new contributions do not break existing functionality, thereby supporting the continuous development and maintenance of open-source software.
- **Promote Reusability:** With better-tested code, open-source projects become more reliable and reusable, encouraging their adoption and adaptation in both academic and commercial contexts.

This project not only aims to streamline the testing process but also enhances the overall health and sustainability of open-source projects, thereby allowing the research community to progress more efficiently.

C. Relation with Software Engineering Research

Automated test generation from natural language specifications is a novel area of research that bridges the gap between traditional software engineering practices and advancements in natural language processing. Work described in ”Automatically Generating Precise Oracles from Structured Natural Language Specifications,” illustrates the potential of this approach. Swami successfully extracts test oracles and generates executable tests from structured natural language, focusing on exceptional behavior and boundary conditions—areas often neglected in manual testing. [1]

D. Key Insight or Idea

The central premise of this project is to harness the capabilities of recent advancements in large language models (LLMs) for the generation of test cases in software engineering. Specifically, we will explore the use of state-of-the-art models like Codex, GPT-J, and GPT-NeoX-20B, which have demonstrated significant potential in understanding and generating code.

Methodological Approach:

- **Model Selection and Integration:** We will begin by selecting several leading LLMs, each known for their strengths in code generation and natural language understanding. The selection will be based on their performance metrics from the latest research, as well as accessibility and integration capabilities.
- **Test Generation Framework:** The chosen models will be integrated into a unified framework that utilizes their code generation capabilities to automatically produce test

cases from existing comments and documentation within software projects.

- **Ensemble Approaches:** Considering the variability in model outputs, we will experiment with ensemble techniques that combine the strengths of multiple LLMs to improve the accuracy and relevance of generated test cases.

E. Assumptions

For the successful implementation and operation of the proposed test generation framework, we assume:

- **Consistent Documentation:** The open-source projects targeted will have reasonably consistent and informative comments and docstrings, which are crucial for accurately generating test cases.
- **Language Compatibility:** The NLP tools and techniques employed are effective across the programming languages prevalent in selected open-source projects.
- **Integration Willingness:** There is willingness among the development community to integrate and actively use this automated testing framework within their development processes.
- **Technological Stability:** The underlying NLP models and test generation algorithms will perform stably across diverse datasets and use cases.

F. Research Questions

The project aims to address the following research questions:

1) How effectively can recent large language models (LLMs) interpret natural language documentation and comments to generate executable test cases?
2) What impact do test cases generated by LLMs have on the overall quality and coverage of test suites in open-source software projects?
3) Which LLMs perform best for test generation tasks, and is there an advantage to using an ensemble of different models?

G. Evaluation Dataset

The evaluation of the proposed framework will be conducted using:

- **Source Selection:** A curated set of open-source projects from platforms such as GitHub, which are known for rich documentation and diverse usage scenarios. Mainly focusing on WEC-GRID.
- **Dataset Characteristics:** Projects chosen will have comprehensive comment and docstring coverage and represent a variety of software types and programming languages to ensure broad applicability of the results.

H. Evaluation Metrics

The success of the test generation framework will be measured by:

- **Test Coverage:** The percentage increase in code coverage achieved by the tests generated by the framework compared to the existing test suites. This will be measured using code coverage tools to quantify the additional lines of code executed by the new tests.
- **Bug Detection:** The number and severity of defects identified by the newly generated tests that were not detected by existing tests. Defects detected will be classified and quantified to assess the impact of the new tests.
- **Precision and Recall:** The accuracy of the test cases in terms of their relevance (precision) and completeness (recall) in covering the described functionalities and behaviors. These metrics will be calculated based on the match between the test cases and the intended functionality as described in the documentation.
- **Developer Feedback:** Qualitative assessments from developers regarding the usability and helpfulness of the automated tests in their daily development workflows. Feedback will be collected through surveys and interviews to gauge satisfaction and practical utility.
- **Integration Ease:** Evaluation of how easily the generated tests can be integrated into existing testing frameworks and CI/CD pipelines, measured by the time and effort required for integration.
- **Test Generation Efficiency:** Time taken from processing input documentation to generating usable test cases. This metric will help evaluate the practicality of deploying the framework in a real-world, agile development environment.

II. Background and Motivation

III. Related Work

IV. Approach

V. Evaluation

A. Dataset

B. Metrics

C. Experiment Procedure

D. Results

VI. Discussion and Threats to Validity

VII. Contributions

References

[1] M. Motwani and Y. Brun, "Automatically generating precise oracles from structured natural language specifications," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 188–199, ISSN: 1558-1225. [Online]. Available: https://ieeexplore.ieee.org/document/8812070