AIProbe

Rahil Mehta
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—**
***Index Terms*—**

I. Introduction

A. Problem statement

Testing Reinforcement Learning (RL) systems effectively pose significant challenges due to their inherent complexity and the dynamic nature of the environments in which they operate. Traditional testing methods often fall short of thoroughly exploring the vast action spaces and diverse scenarios that these systems can encounter, leading to insufficient coverage and potential oversights in the system's ability to handle unexpected or rare situations. Furthermore, the adaptive nature of RL agents, which learn and evolve based on interactions with the environment, adds another layer of difficulty as their behavior can change over time, making consistent testing challenging.

B. Motivation

As reinforcement learning (RL) systems are increasingly deployed in high-stakes environments such as autonomous driving, financial trading, and critical infrastructure management, the imperative to ensure these systems perform reliably under unpredictable conditions cannot be overstated. The complexity and dynamic nature of these applications demand that RL systems not only handle the expected scenarios effectively but also adapt and respond appropriately to unexpected situations. The potential consequences of failures in these areas can be profound, ranging from financial losses to impacts on human safety. Thus, developing robust RL systems that can withstand the variability and uncertainties of real-world environments is crucial. This requires rigorous testing and continuous improvement of their decision-making algorithms to manage and mitigate risks effectively, ensuring that they meet the high reliability standards demanded by these critical sectors

C. Relation with software engineering research

Software engineering encompasses a wide array of techniques dedicated to improving software quality, among which software testing is paramount. Traditional software testing techniques—ranging from unit testing and integration testing to system testing and acceptance testing—serve to verify that software functions as intended and meets all specified requirements. In the context of artificial intelligence (AI) and specifically reinforcement learning (RL), these established methods provide a foundation but also necessitate adaptation to address the unique challenges posed by intelligent systems.

D. Key Insight or Idea

The proposed approach aims to significantly enhance the robustness and adaptability of reinforcement learning (RL) systems by integrating advanced fuzzing techniques, which are specifically designed to generate actionable instructions. These instructions will test the system's capabilities within a given action space and environment. This method focuses on applying a wide range of generated instructions to the RL environment, monitoring both the initial and final states, and analyzing the differences to assess the AI's ability to execute tasks effectively.

E. Assumptions

For the proposed solution to integrate fuzzing techniques into reinforcement learning (RL) systems effectively, several foundational assumptions are made. The primary assumption underlying this approach is that changes in the environment state are driven exclusively by the actions of the RL agent. This assumption is crucial as it simplifies the analysis of the environment's response to the agent's actions, allowing us to directly attribute alterations in the environment state to the instructions processed and actions taken by the AI agent

F. Research questions

Can a fuzzer be employed to not only generate actionable instructions but also help identify tasks that are achievable and not achievable by an AI agent, thereby identifying bugs or limitations in the AI environment?

G. Evaluation Dataset

For the evaluation of our fuzzing-integrated reinforcement learning solution, we will utilize the MiniGrid environment. MiniGrid is a minimalistic grid world package for OpenAI Gym, specifically designed for developing and testing reinforcement learning algorithms. This environment provides a controlled and versatile platform where different scenarios can be programmatically generated, allowing us to apply a wide range of fuzzing-generated instructions to assess the RL agent's adaptability and robustness.

H. Evaluation Metrics

To effectively assess the impact of integrating a fuzzer into a reinforcement learning system for generating instructions, it is essential to employ a set of well-defined and measurable metrics. These metrics are specifically designed to evaluate the fuzzer's ability to enhance the AI agent's performance through diverse, actionable, and extensive instructional sets.

- **Coverage**:
  - *Definition*: Coverage measures the extent to which the fuzzer explores the AI's potential actions within its action space.
- **Actionability**:
  - *Definition*: Actionability assesses the proportion of fuzzed instructions that the AI agent can understand and execute successfully.
- **Diversity**:
  - *Definition*: Diversity evaluates the range of different types of instructions generated by the fuzzer.
- **Consistency**:
  - *Definition*: Consistency measures the reliability and stability of the AI agent's responses to the same or similar fuzzed instructions across multiple instances.

II. Background and Motivation

III. Related Work

[[R2Fix [1] and iFixR [2], use information retrieval-based fault localization (IRFL) that ranks suspicious program statements based on their similarity with bug reports.]]

IV. Approach

V. Evaluation

A. Dataset

B. Metrics

C. Experiment Procedure

D. Results

VI. Discussion and Threats to Validity

VII. Contributions

References

[1] C. Liu, J. Yang, L. Tan, and M. Hafiz, "R2Fix: Automatically generating bug fixes from bug reports," in *IEEE International Conference on Software Testing, Verification and Validation*, 2013, pp. 282–291.

[2] A. Koyuncu, K. Liu, T. F. Bissyandé, D. Kim, M. Monperrus, J. Klein, and Y. L. Traon, "iFixR: bug report driven program repair," in *27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2019, p. 314–325.