Automating Software Repair

Name and Last Name: Amir Arsalan Yavari
Supervisor: Dr. Elham Mahmoudzadeh

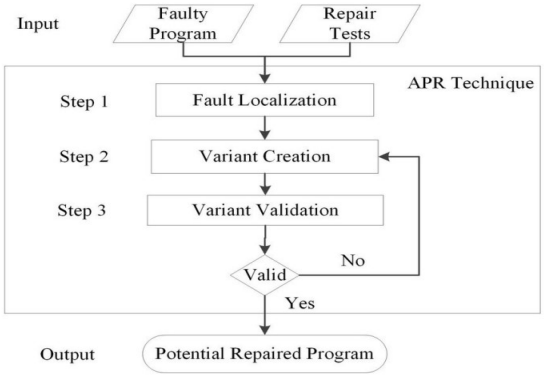
Isfahan University of Technology

1. Introduction

Concept of Problem:

Software testing and repair is a fundamental part of the software production lifecycle. In this phase, software is tested, and software defects and errors are identified and corrected. The increasing complexity of software and the growing number of software applications have made it necessary for us to use automated software repair methods because manual methods are costly and time consuming.

Method of Work:



Inputs: A faulty program and a test suite

Valid State: A suitable state of the program capable of passing the tests

Problem Objective: Generating potentially corrected program

2. Concept of the Reward Repair Model:

➢ **Calculation of Incorrect Performance Rate:**

$$-\infty < R_{no-change} < R_{no-compile} < 0 < R_{compilable} < R_{plausible} < R_{likely\_correct} < 1$$

$$Loss = Loss_{patch\text{-}generator}\,(1-R)$$

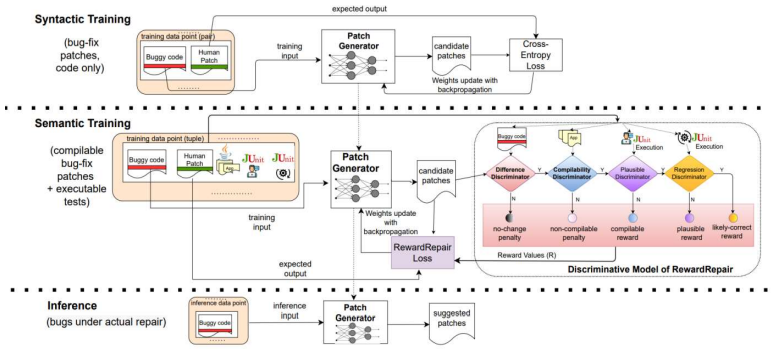
➢ **Two Different type of T5 Transformers:**

T5-small: This is the smallest variant of the T5 model. It has fewer parameters and is suitable for basic text-to-text tasks with relatively small input and output lengths.

T5-base: The base variant is a mid-sized T5 model with more parameters than t5-small. It is a good choice for a wide range of text-to-text tasks and offers a balance between model size and performance.

3. Implementation Method:

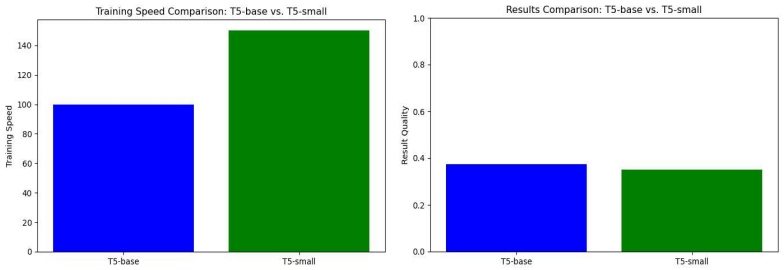
How Reward Repair works:



How to use model and tokenization:

```
model = T5ForConditionalGeneration.from_pretrained('t5-small', output_hidden_states=True)
tokenizer = T5Tokenizer.from_pretrained('t5-small',truncation=True)
tokenizer.add_tokens(['{', '}','<','^','>=','<=','==','buggy:','context:'])
```

4. Results



➢**Better time to train**

➢**The results are almost the same**

5. Conclusions:

The machine learning-based models are still not proficient enough to rectify all software errors. In this project, efforts have been made to reduce the model training time, aiming to quickly provide an initial software corrector that can assist software testers.

Indeed, software testers can utilize automated methods prior to the correction process to address some of the errors and provide suggestions for others.