Automatic Formative Feedback Generation for Programming Tasks Using Program Repair Techniques

Jheison Morales
Rafael Ruiz

Introduction to Data Science and Visualization
Universidad Nacional de Colombia

April 19 2022

Agenda

1. Project Goal and Scope
2. Literature Review
   - Main Concepts
   - Context
   - Timeline
   - Formative Feedback using Automatic Program Repair
3. Case study
   - Typical approach to the program repair task
   - Break-It-Fix-It Approach
4. Datasets
   - Summary Table
   - Propose

UNIVERSIDAD NACIONAL DE COLOMBIA

Project Goal and Scope

Main Goal

Build a machine learning model for automatically generating formative feedback for programming task using a state-of-the-art code repair technique.

Scope

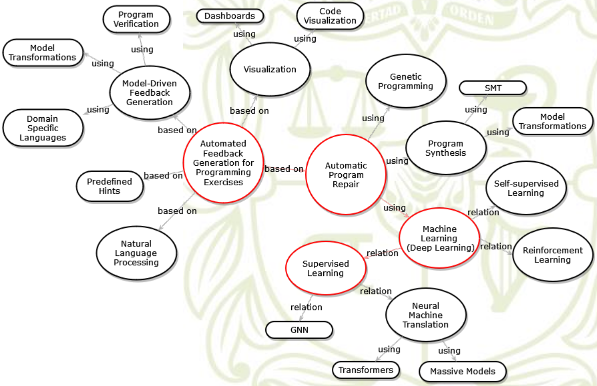
This project is limited to adapting a state-of-the-art repair model for use in generating formative feedback focused on basic python programming tasks.

Main Concepts

- **Formative Feedback:** aimed at helping students to improve their work, is an important factor in learning. [1]
- **Automatic Program Repair:** consists of automatically finding a solution to software bugs without human intervention. [2]

Context

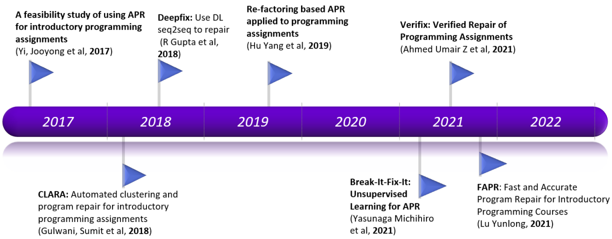
Figura: Relationship between formative feedback and program repair and its most relevant approaches.



Fonte: Based on: [2] [1]

Timeline

Figura: Some interesting approaches using program repair to give formative feedback in the last 5 years.



A feasibility study of using APR for introductory programming assignments (Yi, Jooyong et al, **2017**)

Deepfix: Use DL seq2seq to repair (R Gupta et al, **2018**)

Re-factoring based APR applied to programming assignments (Hu Yang et al, **2019**)

Verifix: Verified Repair of Programming Assignments (Ahmed Umair Z et al, **2021**)

CLARA: Automated clustering and program repair for introductory programming assignments (Gulwani, Sumit et al, **2018**)

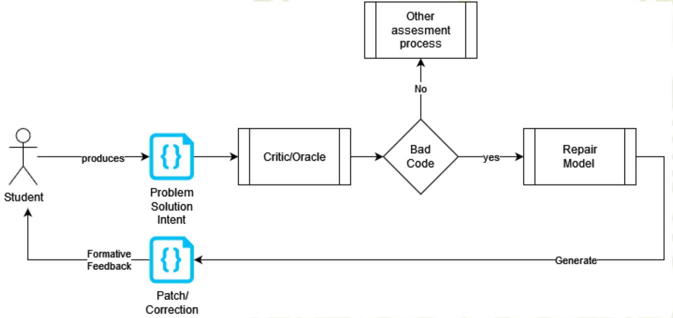
Break-It-Fix-It: Unsupervised Learning for APR (Yasunaga Michihiro et al, **2021**)

FAPR: Fast and Accurate Program Repair for Introductory Programming Courses (Lu Yunlong, **2021**)

Fonte: Based on: [3] [4] [5] [6] [7] [8] [9]

Formative Feedback using Automatic Program Repair

Figura: Simple scheme of the use of automatic repair programs to generate formative feedback.



Typical approach to the program repair task

A common approach to perform learning-based repair tasks is to use neural translation techniques, in which pairs are needed (Buggy Code, Correct Code). To build that kind of data set, there are two options:

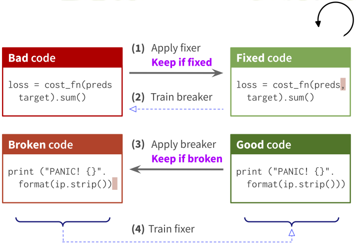
1. Mine repositories and manually identify which commit introduced a bug and which commit resolved it.
2. Create training data consisting of (bad, good) pairs by corrupting good examples using heuristics.

Break-It-Fix-It Approach

Fixers trained on this synthetically-generated data **do not extrapolate well** to the real distribution of bad inputs. BIFI propose:

1. Use the **critic** to check a fixer's output on real bad inputs and add good (fixed) outputs to the training data.
2. Train a **breaker** to generate realistic bad code from good code

Figura: Example of BIFI approach.



```
loss = cost_fn(preds
  target).sum()
```

```
loss = cost_fn(preds,
  target).sum()
```

```
print ("PANIC! {}".
  format(ip.strip())
```

```
print ("PANIC! {}".
  format(ip.strip()))
```

Fonte: Taken from: [7]

Datasets

Identified data sets useful to achieve the objective of this proposal.

Name	Description	Purpose in this project
CodeXBlue [9]	It includes models (based on CodeBERT and GPT) and data sets for diverse task (including program repair)	Benchmark for repair task
MBPP [10]	It consists of around 1,000 crowd-sourced Python programming problems, designed to be solvable by entry level programmers. Each problem consists of a task description, code solution and 3 automated test cases.	Benchmark for feedback generation
BIFI [7]	3 million Python3 snippets from GitHub, divided in two groups: Bad code, Correct code.	Dataset for training/validate models

Propose

This proposal then consists of:

1. Replicating the BIFI work.
2. Using BIFI with the MBPP dataset to evaluate its performance (accuracy, precision, etc.), repairing assignments of basic programming problems.

Outcome

A baseline useful for other Introductory programming assignments repair models.

References I

[1] Hieke Keuning, Johan Jeuring e Bastiaan Heeren. “A Systematic Literature Review of Automated Feedback Generation for Programming Exercises”. Em: **ACM Transactions on Computing Education** 19 (1 jan. de 2019), pp. 1–43. ISSN: 1946-6226. DOI: 10.1145/3231711. URL: https://dl.acm.org/doi/10.1145/3231711.

[2] Martin Monperrus. “Automatic software repair: A Bibliography”. Em: **Proceedings - International Conference on Software Engineering** Part F1371 (1 2018), p. 1219. ISSN: 02705257. DOI: 10.1145/3180155.3182526.

[3] Jooyong Yi et al. “A feasibility study of using automated program repair for introductory programming assignments”. Em: vol. Part F130154. Association for Computing Machinery, ago. de 2017, pp. 740–751. ISBN: 9781450351058. DOI: 10.1145/3106237.3106262.

References II

[4] Sumit Gulwani, Ivan Radiček e Florian Zuleger. "Automated clustering and program repair for introductory programming assignments". Em: **ACM SIGPLAN Notices** 53 (4 jun. de 2018), pp. 465–480. ISSN: 15232867. DOI: 10.1145/3192366.3192387.

[5] Yang Hu et al. **Re-factoring based Program Repair applied to Programming Assignments**. 2019. URL: https://github.com/githubhuyang/refactory.

[6] Rahul Gupta et al. "DeepFix: Fixing Common Programming Errors by Deep Learning". Em: **Proceedings of the AAAI Conference on Artificial Intelligence** 1 (Traver 2017), pp. 1345–1351.

[7] Michihiro Yasunaga e Percy Liang. "Break-It-Fix-It: Unsupervised Learning for Program Repair". Em: (2021). URL: http://arxiv.org/abs/2106.06600.

References III

[8] Umair Z. Ahmed et al. “Verifix: Verified Repair of Programming Assignments”. Em: (jun. de 2021). URL: http://arxiv.org/abs/2106.16199.

[9] Shuai Lu et al. “CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation”. Em: (2021). URL: http://arxiv.org/abs/2102.04664.

[10] Jacob Austin et al. “Program Synthesis with Large Language Models”. Em: (ago. de 2021). URL: http://arxiv.org/abs/2108.07732.

Thanks for your attention

Contact:

`jhmoralesva@unal.edu.co`
`ramruizni@unal.edu.co`

UNIVERSIDAD
NACIONAL
DE COLOMBIA