Break-It-Fix-It: Unsupervised Learning for Program Repair

Aayush Patel, Pranav Patel, Kalp Shah and Vatsal Shah

Abstract

This work explores the "Break-It-Fix-It" (BIFI) framework, which introduces an unsupervised learning approach to automatic program repair (APR). BIFI leverages neural models to generate buggy programs and subsequently fixes them. The dual-stage process ensures that the repair mechanism learns robust representations for fixing errors while mitigating dataset biases. This abstract outlines the problem, dataset, evaluation metrics, results, and key challenges addressed during the research.

Keywords

Program Repair, Unsupervised Learning, Neural Networks, Software Engineering, Information Retrieval

1. Problem

We are given a **Critic** that assesses the quality of input code. Specifically, code is classified as *bad* if it contains at least one error, and as *good* if it is error-free. Our goal is to train a **Fixer** that repairs *bad code* into *good code*, ensuring it satisfies the Critic. To achieve this, we use unlabeled data and the Critic to learn the Fixer, bypassing the need for manually curated paired datasets.

Traditional supervised learning approaches rely on supervised learning and annotated paired datasets of *bad code* and corresponding *good code* to train models. However, these datasets are both expensive to create and inherently limited in scale and diversity. Real-world programming errors are highly complex and context-dependent, making it infeasible to curate datasets that comprehensively represent the full distribution of errors encountered in practice. Furthermore, synthetic paired datasets, often created by random perturbations (e.g., dropping tokens or adding errors), fail to accurately reflect real-world errors, leading to poor performance when applied to real-world scenarios.

The unsupervised approach circumvents these challenges by leveraging the Critic to generate training pairs dynamically. By iteratively generating and refining training data, we adapt the Fixer to real-world error distributions, improving its ability to repair code effectively.

2. Dataset

The **GitHub-Python** dataset is a large-scale collection of Python code snippets, specifically curated to facilitate the task of unsupervised program repair. It consists of approximately **3 million (3M)** code snippets. Below is a detailed description of the dataset:

Good and Bad Code

The dataset is unpaired and consists of:

- **Good Code:** Approximately **3M snippets** of syntactically correct Python code.
- **Bad Code:** A total of **38,000 snippets** containing syntax errors such as unbalanced parentheses, indentation errors, and other parse issues.

Introduction to Information Retrieval (IT550), Autumn 2024, DAIICT, India

aayush.patel@daiict.ac.in (A. Patel); 202103040@daiict.ac.in (P. Patel); 202103003@daiict.ac.in (K. Shah); 202103022@daiict.ac.in (V. Shah)

Dataset Columns

Each code snippet in the dataset is represented using the following fields:

- `code_string`: The raw Python code snippet as a string.
- `code_toks_joined`: A tokenized version of the code snippet.
- `anonymize_dict`: A dictionary mapping anonymized tokens (e.g., `<STRING>`) to their original representations in the code.
- `err_obj`: An object describing the specific syntax error detected by the AST parser.

Train-Test Split

- **15,000 snippets of bad code** are held out as the **test set**.
- The remaining **23,000 snippets of bad code** are used for training the Break-It-Fix-It (BIFI) framework.

3. Evaluation Metrics

The evaluation metric used to assess the performance of the Fixer model is **Repair Accuracy**. It measures the proportion of bad code examples from the test set that are successfully repaired by the Fixer. A repair is considered successful if the repaired code is classified as *good* by the Critic.

$$\text{Repair Accuracy} = \frac{\left|\{x \mid x \in D_{\text{bad}}^{\text{test}}, c(f(x)) = 1\}\right|}{\left|D_{\text{bad}}^{\text{test}}\right|}$$

In simpler terms, **Repair Accuracy** measures the fraction of bad examples in the test set that are successfully fixed by the Fixer.

4. Results

The following table compares the repair accuracy achieved by our implementation of the Break-It-Fix-It (BIFI) framework with the accuracy reported in the original paper:

Method	Our Accuracy (%)	Paper's Accuracy (%)
Initial Fixer (R0)	54.2	62.0
Backtranslation (R2)	68.3	80.1
BIFI (R2)	74.8	90.5

Table 1
Comparison of repair accuracy between our implementation and the original paper.

The accuracy achieved by our implementation is lower than the paper's reported results across all stages (R0, R2). The main reason for this discrepancy is the use of a *smaller number of layers* in our Fixer and Breaker models during training, compared to the original model used in the paper. A smaller model architecture may result in reduced capacity to learn the complex transformations required for code repair, leaading to lower overall accuracy. Additionally, hyperparameter tuning and data preprocessing differences might have contributed to the performance gap.

5. Key Challenges and Learnings

- **Challenge:** Generating realistic and diverse buggy programs without introducing biases.
- **Learning:** Incorporating adversarial training helped improve the robustness of the "Breaker" model.

- **Challenge:** Ensuring scalability while training neural models on large code datasets.
- **Learning:** Optimized model architectures and distributed training significantly reduced computation overheads.

6. Conclusion

The ”Break-It-Fix-It” framework presents a novel and effective approach to unsupervised learning for program repair. By addressing key limitations of existing APR techniques, it paves the way for future research in scalable and self-supervised program repair methodologies.

References

[1] Michihiro Yasunaga and Percy Liang, *Break-It-Fix-It: Unsupervised Learning for Program Repair*, CoRR, vol. abs/2106.06600, 2021. Available: https://arxiv.org/abs/2106.06600.