RLInFix: A Reinforcement Learning Approach for Automatically Repairing Beginners' Program Inputs

Ni Mu[*]
Department of Computer Science
Southeast University
Nanjing, Jiangsu, China
muni@seu.edu.cn

Xianglong Kong
Department of Computer Science
Southeast University
Nanjing, Jiangsu, China
xlkong@seu.edu.cn

Abstract

This paper presents RLInFix, a framework for automatically fixing program inputs for beginner programmers using Reinforcement Learning and the Attention model for state encoding. RLInFix is inspired by InFix[2], which also repairs the input data to make programs run successfully, instead of focusing on the way of debugging program codes as most works do. Our framework is based on Reinforcement Learning (RL), because the whole iterative repairing process is very similar to a typical RL trajectory. Inspired by the five templates and five mutations for input repairing in InFix[2], six parameterized actions are used in our framework, which is strong and generalizable. The reward for RL is determined by the correctness of the input. The popular Attention model is used to encode the program and error messages, for programs are similar to natural language which has context and needs the "attention" of human. This novel framework is proposed to broaden the idea of software fault research, as well as provide methodology for related code implementation. Thus, implementation for RLInFix is not provided in this work.

Keywords: software faults; input repair; Deep Learning; Reinforcement Learning; Attention.

1 Introduction

Though online education resources are becoming more and more abundant, beginners of various programming languages still often feel frustrated in the process of getting started. The time of debugging a single error is positively related to the frustration of beginners. Currently, most of the current works focus on how to debug user's code. However, take Python as an example: according to the data from Python Tutor[3], a surprisingly large proportion of errors is input related (e.g., entering 1,2 instead of 1.2). Therefore, for the popularity of programming technique, it is significant to provide support or hints for the beginners' debugging process, from the perspective of input.

Endres et al. made some meaningful exploration in this direction. They present InFix[2], a randomized template-based approach for automatically fixing erroneous inputs for beginner programmers. Taking advantage of input error patterns, InFix can iteratively repair the inputs. InFix was evaluated on 25,995 unique input-related scenarios and repaired 94.5% of input errors with average time of 1.2s per input.

We try to utilize Machine Learning and Reinforcement Learning to repair program inputs, hoping they can lead to better results and generalization. Since 2010, in software fault localization and repair, Data Mining and Machine Learning works increase, and commonly-used technologies include Decision Tree, Naive Bayes, Support Vector Machine, etc[1]. In the past two years, Deep Learning techniques

[*]Student Number: 09019106

have been applied to software fault localization and repair[1]. The commonly used technologies include Convolutional Neural Network (CNN), Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), etc. Most of these works base on the idea of tokenization or neural embedding[1], where Machine Learning and Deep Learning technologies act like powerful function fitters. Thus, Machine Learning and Deep Learning are suitable for these tasks. Nevertheless, Reinforcement Learning is a framework for interacting with the environment and solving difficult tasks, which requires corresponding task structures[9]. Therefore, Reinforcement Learning works in software fault localization and repair are rarely observed.

We found two typical cases of Reinforcement Learning work, both in the field of software testing. One of them [8][4] use curiosity driven Reinforcement Learning to model the interaction between the tester (agent) and the application (environment), effectively improving the code coverage of the test. The other one [6][5] use Reinforcement Learning to automatically sort the test cases of software testing over a long period of time, take the historical execution information as the observation, and uses reward mechanism of Reinforcement Learning to continuously optimize the sorting algorithm. However, no work is found to utilize Machine Learning and Reinforcement Learning to repair program, especially to repair program inputs.

This paper presents RLInFix, a Reinforcement Learning framework for automatically fixing erroneous program inputs for beginners. RLInFix repairs input data rather than source code, requires no test cases, and requires no special annotations. Based on the work of InFix[2], we 1. model the iterative fixing process of InFix as an RL process, 2. take advantage of beginners' input patterns to automatically create powerful and generalizable input repairs, and 3. use the Attention model to efficiently encode both the program codes and error messages.

In summary, our main contributions are as follows:

- Propose RLInFix framework;
- Provide a more specific implementation idea of RLInFix for Python.

Note that this framework is not only applicable to python. The ideas of the action patterns and the Attention-based encoding are universal, and the rationality of Python has been proved by InFix[2]. Our novel framework is proposed to broaden the idea of software fault research, as well as provide methodology for related code implementation. Thus, implementation for RLInFix is not provided in this work.

The following parts of the paper are organized as follows: Section 2 introduces the preliminaries, including InFix[2], Reinforcement Learning, and the Attention model; Section 3 describes the general structure of RLInFix, as well as how it can be implemented for Python; Section 4 provides a discussion on possible limitations of RLInFix (only "possible" limitations for no experiment is done); Section 5 is the conclusion part.

2 Preliminaries

2.1 InFix

With the assumption that the program is correct, InFix[2] focuses on repairing the inputs that cause errors, rather than debugging program codes as most works do. An essential observation is that, many inputs that cause errors have similar reasons, and these reasons can be generalized to specific patterns. Taking Python as an example, according to the dataset from Python Tutor[3], the errors related to integer input format take 34.42% in total, the list packing format 9.09%, the float input format 8.62%, and the list index problem 18.56%. Also, an erroneous input may contain more than one patterns, which need to be fixed iteratively one by one. Therefore, a fixing algorithm using random search is proposed. For an error input I, we first check that if any template T can be fitted. If so, $I \leftarrow T(I)$. Else, take a random mutation M, that is, $I \leftarrow M(I)$. This step is done repeatedly until I goes correct.

InFix is surprisingly effective. It can successfully repair 94.5% of input-related errors in Python Tutor dataset, with high quality rated by humans. It is also efficient, spending a median time of 0.88s and an average time of 1.23s per error. However, InFix may lose the semantics of test cases. This involves the definition of correctness in the random search iterative fixing process. For example, a beginner wants to enter an integer 66, but he enters "66" with quotation marks. Fixing "66" to 66 or fixing it to

0 can be both executed successfully, but only the former one doesn't lose the semantics. The authors of InFix puts forward a related assumption: beginners only want to successfully run the program, and they do not care about the semantics.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a kind of general idea or framework, where an agent learns to complete tasks through interaction with the external environment. Specifically, as Figure 1 shows, in each round of interaction, the agent obtains the observation of the current environment, takes an action according to the observation, and obtains the feedback reward of the environment and the updated observation.

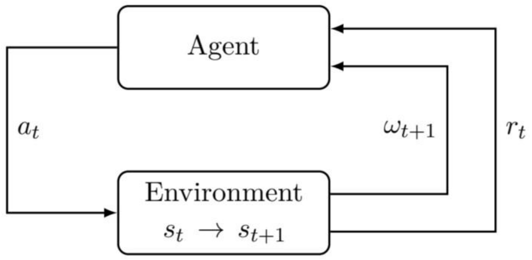

Figure 1: The framework of Reinforcement Learning

The agent interacts with the environment, until we can make sure the task is completed successfully or it fails. This process is called an episode. The objective function of Reinforcement Learning is to maximize the total rewards in the whole episode.

2.3 The Attention Model

The Attention model is a kind of neural network structure, which was first proposed by Vaswani et al. [7], and is applied to sequence modeling. It is widely used in Computer Vision and Natural Language Processing because of its unique attention mechanism, which is similar to the mechanism of people understanding long texts as well as images. In this work, it is used to encode the programs and error messages.

3 Method

3.1 The RL Framework

The RLInFix framework is based on Reinforcement Learning (RL), because the whole iterative repairing process is very similar to a typical RL trajectory. The basic elements of RL are as follows:

Environment: The whole system is needed to be divided into environment and agent. Generally, in RL, the part we can control through commands is divided into agents, and the other part we do not know and may work with randomness is the environment. In RLInFix, programs and compilers or interpreters are regarded as the environment. The output of compilers or interpreters (such as error messages) is feedback from the environment.

Agent: In RLInFix, the part of user inputs is treated as agents. The action we take is to change the inputs according to environment feedback, in order to make the program run successfully as soon as possible.

Observation: When we humans are modifying the error input, we will read through the source code, current input, and error message. The observation of RLInFix also includes these parts. Note that we humans also apply some kinds of attention mechanisms when reading programs, such as tracing which paragraphs of codes affect a variable, and locating the buggy line of code according to error messages. Therefore, the Attention architecture is used to encode the observation. Section 3.2 provides a detailed description in terms of repairing beginners' Python inputs.

Action: Inspired by the 5 templates and 5 mutations in InFix[2], we define 6 parameterized actions to modify the current inputs. Section 3.3 provides a detailed description in terms of repairing beginners' Python inputs.

Reward: The objective of RL is to maximize the total reward for the whole episode. Therefore, if the current input is not correct, we give the agent a penalty of -1 as the reward. Otherwise, the reward is 0.

Done (episode end): When the current input is correct, the episode ends (and the reward is 0).

The whole architecture is shown in Figure 2.

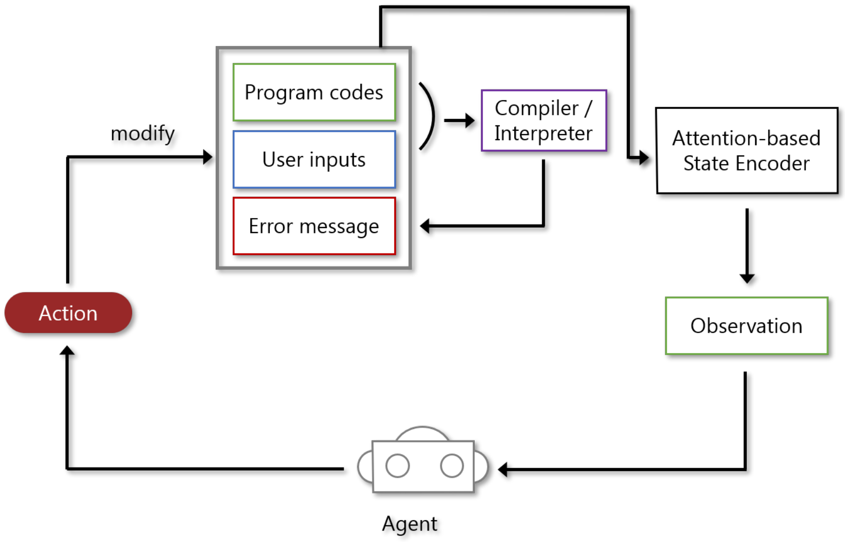

Figure 2: The whole architecture of the RLInFix framework

3.2 The Attention-based State Encoder

This subsection provides a detailed idea for the implementation of RLInFix in the scenario of Python input fixing. The Attention model is utilized to extract the information related to the error, including 1. the location of error occurrence, 2. the type and details of the error, 3. which inputs caused the error, and 4. how the error-relevant variables were processed in the previous program sentences.

In the Python scenario, a typical error message is as follows:

```
Traceback (most recent call last):
    File "E:\path\to\the\program\codes.py", line 11
ValueError: invalid literal for int() with base 10: '3.3'
```

It includes 1. the line number of the error, 2. the error type, and 3. details of the error, which can be all extracted through a hard-coded process. After parsing the error message, the Attention model is used to distill the relationship between the error line of code and the previous program sentences. At the same time, the Attention model is also applied to encode user inputs. The whole structure is shown in Figure 3.

3.3 Actions

We define six parameterized actions as Table 1.

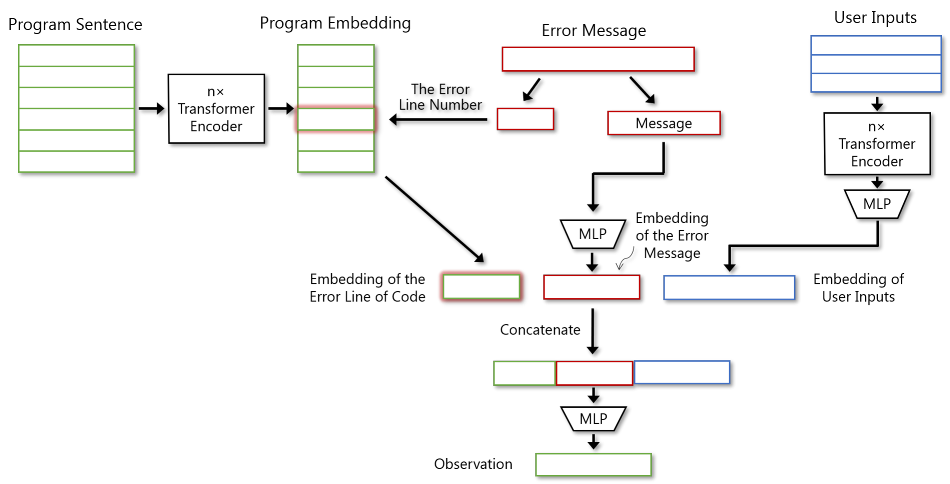

Figure 3: The structure of the state encoder

Table 1: The six actions of RLInFix

Action	Description	Parameters
Insert a line	Insert a new line at the end of the input	/
Delete a line	Delete the last line of the input	/
Insert a token	Insert a new token at the end of a specific line of the input	the line number
Delete a token	Delete the last token of a specific line of the input	the line number
Modify a token to integer	Modify a a specific token to a random integer of [-1, 10] of a specific line of the input	the order of the token, the line number
Modify a token to float	Modify a a specific token to a random float of [-1.0, 10.0] of a specific line of the input	the order of the token, the line number

3.4 Training methods

When training the RL model, the tricks of behavior cloning and self imitation may help. Specifically, we can collect cases of users' successful repairing processes as dataset of behavior cloning, and also let agents repeatedly learn their own successful cases (self imitation).

4 Possible Limitations

Correctness and the loss of semantics Whether for InFix[2] or RLInFix, semantics may be lost in the fixing process. As Section 2.1 shows, the loss of semantics is related to the definition of correctness. Adding a similarity term to the correctness judgment with the original input may alleviate this kind of loss.

"Suicide" shortcut for the agent During the training process, the agent may take some kinds of "suicide" behaviors, such as erasing all input lines one by one. When the input is empty, the program may just run successfully, which encourages the agent to give up mining patterns and take the "suicide" shortcut. Similarly, adding a similarity term to the correctness judgment with the original input may alleviate this kind of loss.

5 Conclusion

This paper presents RLInFix, a Reinforcement Learning framework for automatically fixing erroneous program inputs for beginners. RLInFix repairs input data rather than source code, requires no test cases, and requires no special annotations. Based on the work of InFix[2], we 1. model the iterative fixing process of InFix as an RL process, 2. take advantage of beginners' input patterns to automatically create powerful and generalizable input repairs, and 3. use The Attention model to efficiently encode both the program codes and error messages.

References

[1] Iqra Batool and Tamim Ahmed Khan. Software fault prediction using data mining, machine learning and deep learning techniques: A systematic literature review. *Computers and Electrical Engineering*, 100:107886, 2022.

[2] Madeline Endres, Georgios Sakkas, Benjamin Cosman, Ranjit Jhala, and Westley Weimer. Infix: Automatically repairing novice program inputs. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 399–410. IEEE, 2019.

[3] Philip J Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 579–584, 2013.

[4] Minxue Pan, An Huang, Guoxin Wang, Tian Zhang, and Xuandong Li. Reinforcement learning based curiosity-driven testing of android applications. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2020, page 153–164, New York, NY, USA, 2020. Association for Computing Machinery.

[5] Tingting Shi, Lei Xiao, and Keshou Wu. Reinforcement learning based test case prioritization for enhancing the security of software. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 663–672, 2020.

[6] Helge Spieker, Arnaud Gotlieb, Dusica Marijan, and Morten Mossige. Reinforcement learning for automatic test case prioritization and selection in continuous integration. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2017, page 12–22, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[8] Thi Anh Tuyet Vuong and Shingo Takada. *A Reinforcement Learning Based Approach to Automated Testing of Android Applications*, page 31–37. Association for Computing Machinery, New York, NY, USA, 2018.

[9] Cody Watson, Nathan Cooper, David Nader Palacio, Kevin Moran, and Denys Poshyvanyk. A systematic literature review on the use of deep learning in software engineering research. *ACM Trans. Softw. Eng. Methodol.*, 31(2), mar 2022.