Self-Edit: Fault-Aware Code Editor for Code Generation

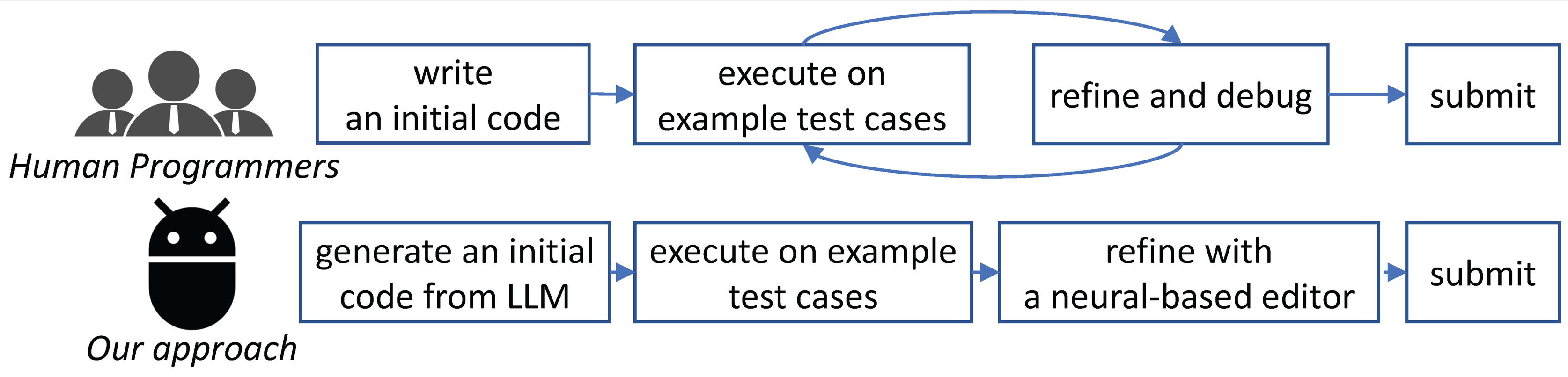
Kechi Zhang, Zhuo Li, Jia Li, Ge Li*, Zhi Jin*

Key Lab of High Confidence Software Technologies (Peking University), Ministry of Education;

School of Computer Science, Peking University, Beijing, China

Introduction

- Existing Large Language Models (LLMs) have low accuracy and pass rates when applied to competitive programming tasks.
- To improve LLMs' performance in this area, we take inspiration from the human programming process, where programmers refine their code based on test results.
- We aim to improve the quality of code generated by LLMs through editing utilizing execution results, taking advantage of the direct feedback provided by error messages.



Our approach is inspired by the problem-solving process of human programmers.

Problem Description: *APPS-test-673*

```
Find the smallest integer x greater than n,
so it is divisible by the number k.
The input only line contains two integers
n and k (1 ≤ n, k ≤ 10^9).
-----Examples-----
Input
5 3
Output
6
```

example test case

GPT3 Output:

```
n, k = map(int, input().split())
if n % k == 0:
    print n + k
else:
    print n + k - (n % k)
```

Error Message:
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(n + k)?

An example of competitive programming tasks and its corresponding GPT3 output

Methodology

We propose Self-Edit, a generate-and-edit approach for Large Language Models (LLMs) that produces high-quality code for competitive programming tasks. Self-Edit consists of three steps:

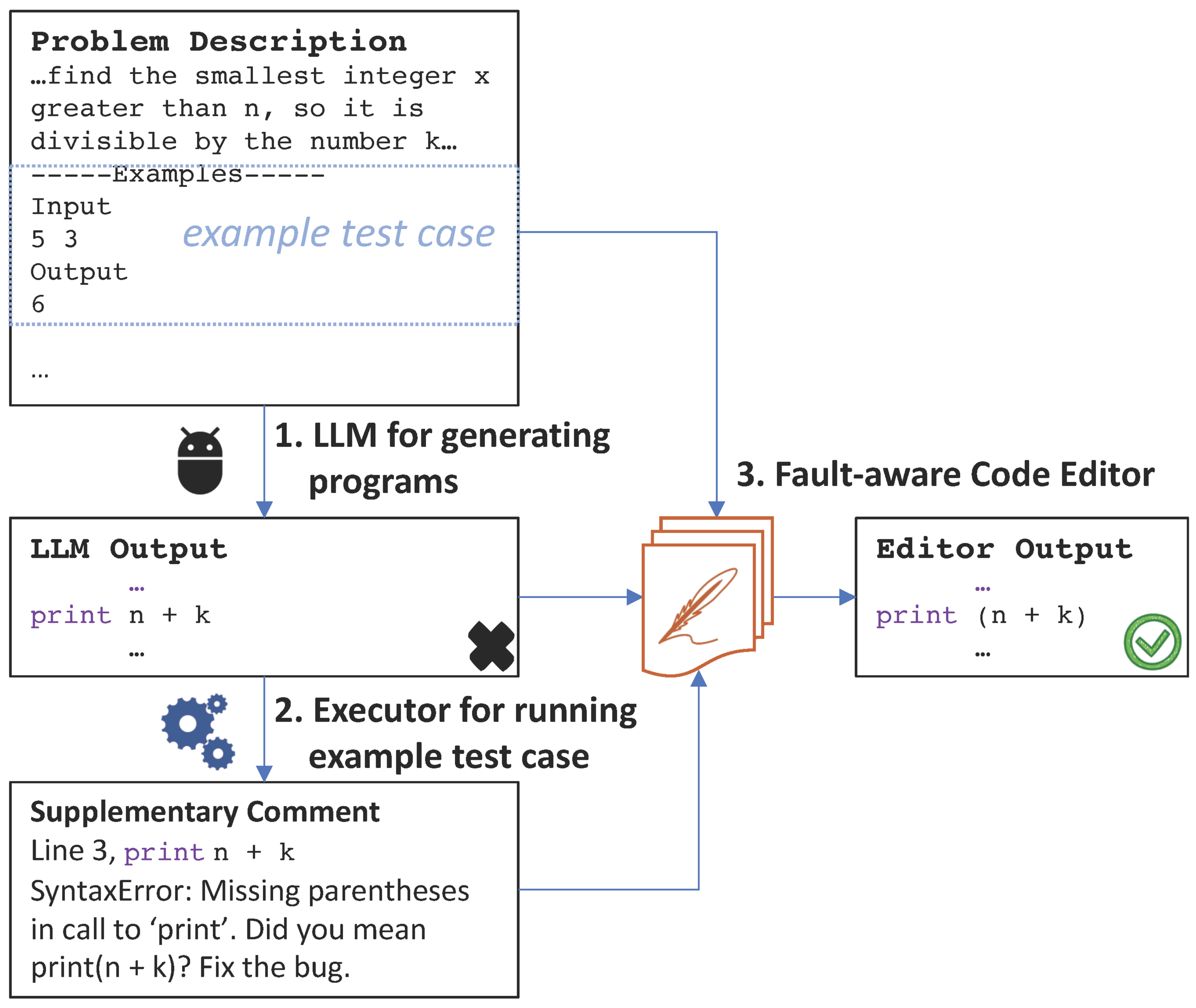
- We use LLMs as a black-box generator to create an initial code based on the problem description.
- We execute example test cases to obtain test results and construct supplementary comments.
- We train a fault-aware code editor model using the problem description, generated code, and supplementary comment to refine the initial code.

```
Comment 1:
Pass the example test case.

Comment 2:
Template: Wrong Answer with input: <input>. Expected
output is <output_1>, but generated output is <output_2>.
Rewrite the code.
Example: Wrong Answer with input: 2 5 3. Expected output
is 1, but generated output is 0. Rewrite the code.

Comment 3:
Template: Line <lineno>, <line_content>, <error_msg>.
Fix the bug.
Example: Line 2,
return len([i for i in str(i**2) for i in range(n+1) if
i == str(d)])
NameError: name 'i' is not defined. Fix the bug.
```

Example Supplementary Comments in different situations



Pipeline of our Self-Edit approach

Experiments

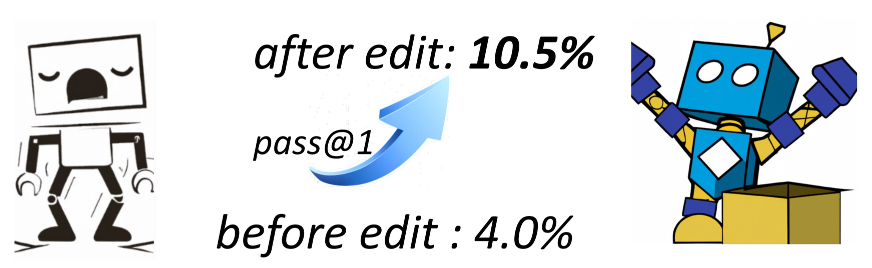
Datasets: *APPS, HumanEval*

Why you should choose self-edit?

- ***A much smaller editor (110M) can even boost the 175B GPT3 for complex code generation tasks!***

  Compared to directly generating from LLMs, our self-edit can improve the average of pass@1 by:
  - **89%** on APPS-dev,
  - **31%** on APPS-test
  - **48%** on HumanEval

  over nine popular code generation LLMs with parameter sizes ranging from 110M to 175B.



An example improvement of pass@1 for *GPT-Neo-1.3B* on *APPS-dev*

- Compared to other post-processing methods, our method demonstrates superior ***accuracy and efficiency***.

		APPS-dev		APPS-test	
Setting	Samples	@1	@5	@1	@5
base model		4.0	10.9	0.14	0.74
+ ranker†	100	8.0	15.1	0.3	1.1
+ editor	{1,5}	**10.5**	**18.6**	**0.68**	**1.38**

† The results are copied from the original paper.

Comparison with a state-of-the-art reranking method *CodeRanker*
Base model: GPT-Neo-1.3B-finetuned

➢ Our approach can achieve better performance with less sample budgets.

- Our self-edit framework can be ***extended using in-context learning*** without additional training

Benchmark		pass@1	pass@5	sol@5
APPS-test	before	7.48	15.94	1876
	after	**8.94**	**17.12**	**2214**
HumanEval	before	34.76	60.98	288
	after	**39.63**	**64.63**	**331**

In-context learning self-edit with *text-davinci-002*

➢ We will explore strategies to efficiently utilize the in-context learning capabilities of LLMs in our self-edit framework in future work.