Large Language Models for Code Intelligence Tasks

LYU2301

Supervisor: Professor Michael R. Lyu

Presenter: Canran Liu, Xingyun Ma

香港中文大學
The Chinese University of Hong Kong

Contents

1. Introduction

- Recap
- Updates

Introduction - Recap

- What we did last term?
  - ➢ Proposed timeEval benchmark.

- APPS Dataset --> timeEval Dataset
  - 110 Problems: Python
  - Canonical solution for each problem
  - Test cases for each question

- A framework for automated measurement of code efficiency

Introduction - Recap

- What we did last term?
  - Proposed timeEval benchmark.
  - On our benchmark, we did several experiments to test the performance of different methods in terms of code efficiency.

Introduction - Recap

- What we did last term?
  - Proposed timeEval benchmark.
  - On our benchmark, we did several experiments to test the performance of different methods in terms of code efficiency.
  - Tried several frameworks to improve the efficiency of generated code.

Experiment	Pass Rate	Wrong Rate	Timeout Rate	%Opt	%Sp
Self-refinement + One-shot	58.9	22	19.1	25.5	35.4
Self-refinement + One-shot +CoT	35.8	55.4	8.8	60.0	84.8
Self-refinement + One-shot +CoT + Test cases	40.8	49.9	9.3	53.6	72.5

Introduction - Update

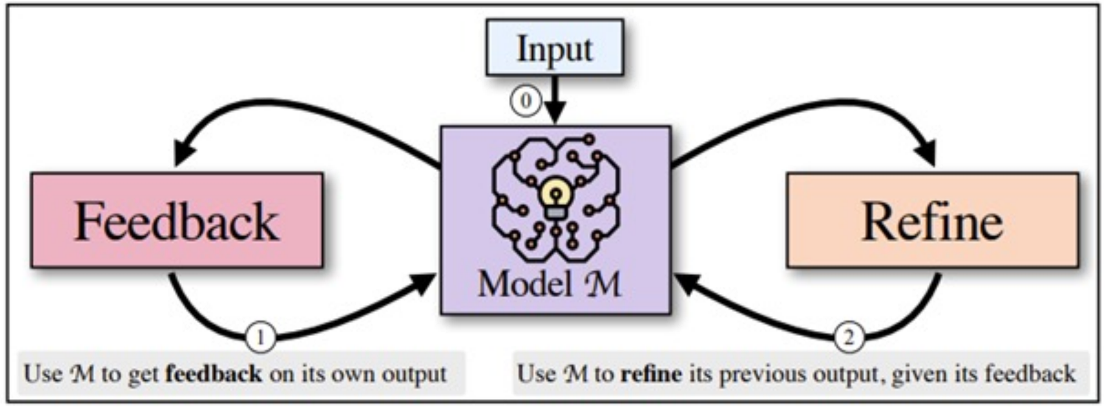
- What we updated this term?
  - Updated timeEval benchmark.

- APPS Dataset & CodeContests --> timeEval Dataset
  - 111 Problems: Python, C++, Java
  - Canonical solution for each problem
  - Low-time-complexity solution for each problem
  - Test cases for each question

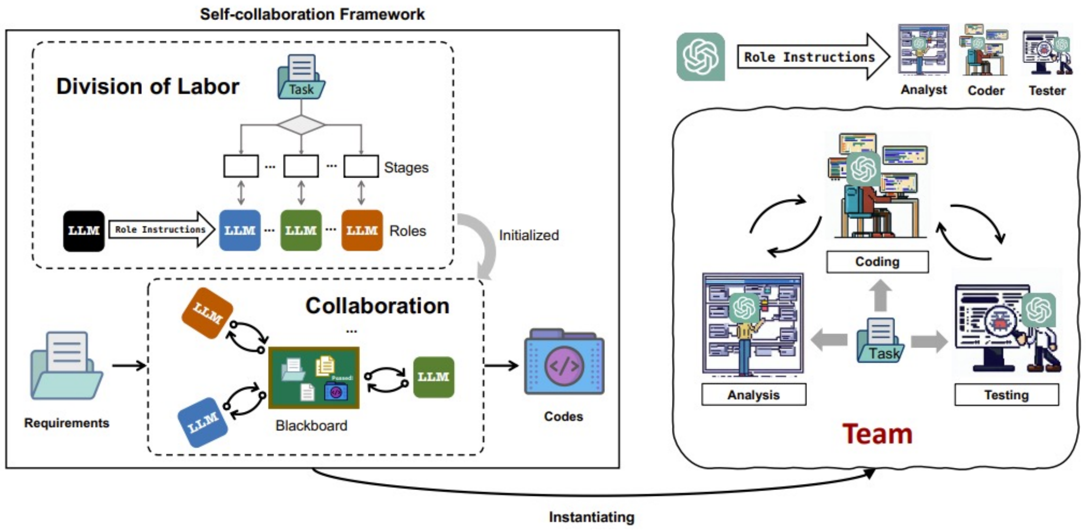
- A framework for automated measurement of code efficiency:
  - Redesign the metrics

Introduction - Update

- What we updated this term?
  - ➢ Updated timeEval benchmark.
  - ➢ On our updated benchmark, we did empirical studies of self-refine and multi-agent collaboration to test the efficiency of generated code.



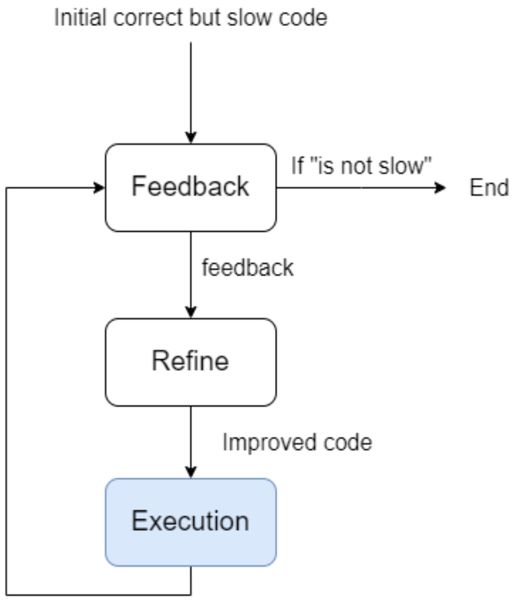
"SELF-REFINE: Iterative Refinement with Self-Feedback"



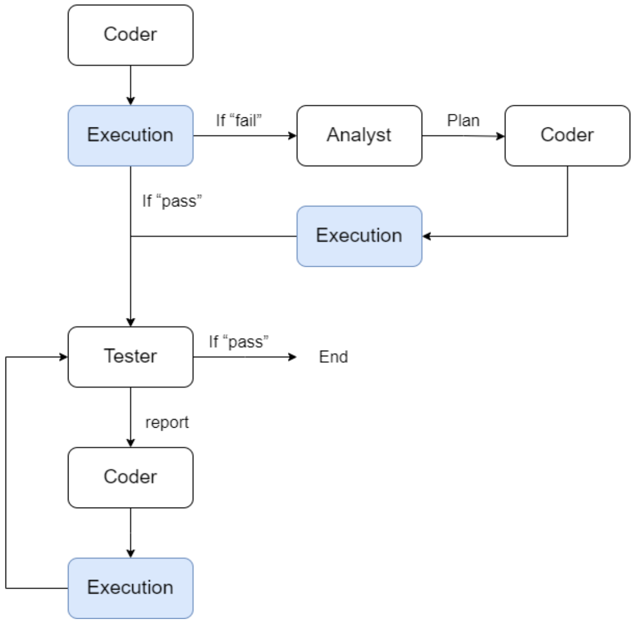
"Self-collaboration Code Generation via ChatGPT"

Introduction - Update

- What we updated this term?
  - Updated timeEval benchmark.
  - On our updated benchmark, we did empirical studies of self-refine and multi-agent collaboration to test the efficiency of generated code.
  - Proposed our frameworks to improve the efficiency of generated code.



Self-Refine-Executor



Multi-Agent-Executor

Contents

2. Analyzing existing datasets

Analyzing existing datasets for Code Generation

Name	Time	Author	Language	Source	Difficulty	#Train	#Test	#Valid	Avg Test Cases	Avg Problem Words	Avg LOC Solution	Paper	Code
APPS	20 May 2021	Dan Hendrycks (UC Berkeley) et al.	Python	Codeforces	competition	5000	5000	-	13.2	293.2	18.0	Measuring Coding Challenge Competence With APPS	Github
HumanEval	7 Jul 2021	OpenAI	Python	-	Simple Software Interview		164	-	7.7	23	6.3	Evaluating Large Language Models Trained on Code	GitHub
MBPP	16 Aug 2021	Google Research	Python	-	entry-level	374	500	90	3.0	15.7	6.7	Program Synthesis with Large Language Models	Github
CodeContests	8 Feb 2022	DeepMind	Python2&3, C++, Java	CodeChef, Codeforces, HackerEarth, AtCoder, Aizu	Competition	13328	165	117	95.9	-	59.8	Competition-Level Code Generation with AlphaCode	Github
DS-1000	18 Nov 2022	Yuhang Lai (HKU) et al.	Python	StackOverflow	-	-	1000	-	1.6	140	3.6	DS-1000: A Natural and Reliable Benchmark for Data Science Code Generation	Github
HumanEval+	2 May 2023	Jiawei Liu et al.	Python	-	Simple Software Interview	-	164	-	774.8	23	6.3	Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation	Github
ClassEval	3 Aug 2023	Xueying Du (FDU) et al.	Python	-	class-level	-	100	-	33.1	-	45.7	ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation	GitHub

Contents

3. Dataset Processing & Enhancement

Dataset Processing & Dataset Enhancement

Old dataset: Python → New dataset

- Python
- C++
- Java

Dataset Processing & Dataset Enhancement

Code_contests dataset

- 13,610 coding problems in total.
- More than 30 test cases for each problem
- There are more than 30 ground truth solutions for each problem in each language.
- Support Python2, Python3, Java, and C++

Dataset Processing & Dataset Enhancement

Dataset cons: Too difficult.

Rank Model Test Set Test Set Test Set Val Set Val Set Paper Code Result Year Tags

Site	URL	Source
Aizu	https://judge.u-aizu.ac.jp	CodeNet
AtCoder	https://atcoder.jp	CodeNet
CodeChef	https://www.codechef.com	description2code
Codeforces	https://codeforces.com	description2code and Codeforces
HackerEarth	https://www.hackerearth.com	description2code

Dataset Processing & Dataset Enhancement

Step 1:

Find the canonical solution among the first 20th ground truth solutions in the dataset:

solution_result_cpp > 00022_result.txt

```
solution_1.cpp:
    Results: [False, False, False, False, True, False, False, True, True, True, True, True, True, False, F
    Outputs: ['8\n5 3 -3 4 -4 1 -1 2\n3\n1 -1 2\n5\n5 4 3 2 1\n', '8\n5 3 -3 4 -4 1 -1 2\n3\n1 -1 2\n5\n5
    Passed tests: 18
    Failed tests: 12
    Execution time: 0.64 seconds

solution_2.cpp:
    Results: [False, False, False, False, False, False, False, False, False, False, False, False, False, F
    Outputs: ['8\n5 1 -1 3 -3 4 -4 2\n1\n2\n5\n5 4 3 2 1\n', '8\n5 1 -1 3 -3 4 -4 2\n1\n2\n5\n5 4 3 2 1\n'
    Passed tests: 0
    Failed tests: 30
    Execution time: 0.84 seconds

solution_3.cpp:
    Results: [False, False, False, False, False, False, False, False, False, False, False, False, False, F
    Outputs: ['8\n2 1 -1 3 -3 4 -4 5\n3\n1 -1 2\n5\n5 4 3 2 1\n', '8\n2 1 -1 3 -3 4 -4 5\n3\n1 -1 2\n5\n5
    Passed tests: 0
    Failed tests: 30
    Execution time: 0.67 seconds

solution_4.cpp:
    Results: [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
    Outputs: ['8\n2 3 -3 4 -4 1 -1 5\n3\n1 -1 2\n5\n5 4 3 2 1\n', '8\n2 3 -3 4 -4 1 -1 5\n3\n1 -1 2\n5\n5
    Passed tests: 30
    Failed tests: 0
    Execution time: 0.66 seconds

solution_5.cpp:
    Results: [False, False, False, False, False, False, False, False, False, False, False, False, False, F
    Outputs: ['8\n2 1 -1 4 -4 3 -3 5\n3\n1 -1 2\n5\n5 4 3 2 1\n', '8\n2 1 -1 4 -4 3 -3 5\n3\n1 -1 2\n5\n5
    Passed tests: 0
    Failed tests: 30
    Execution time: 0.63 seconds
```

Dataset Processing & Dataset Enhancement

Step 2:

Test the generated solution:

```
test_result_java > 00008_result.txt
canonical_solution.java:
    Results: ['True', 'True', 'True', 'True', 'True', 'True', 'True', 'True', 'True', 'True', 'True', 'True', '
    Outputs: ['2\n', '1\n', '0\n', '4\n', '0\n', '6\n', '5\n', '4\n', '1\n', '9\n', '3\n', '6\n', '1\n', '2\n',
    Passed tests: 179
    Wrong answers: 0
    Time limit exceeded: 0
    Execution times: ['0.050', '0.043', '0.044', '0.042', '0.045', '0.043', '0.043', '0.043', '0.043', '0.043',
    Total time: 7.51 seconds

gen_solution.java:
    Results: ['False', 'False', 'True', 'False', 'True', 'False', 'False', 'False', 'False', 'False', 'False',
    Outputs: ['0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n', '0\n',
    Passed tests: 30
    Wrong answers: 149
    Time limit exceeded: 0
    Execution times: ['0.080', '0.088', '0.082', '0.096', '0.092', '0.094', '0.091', '0.078', '0.072', '0.064',
    Total time: 11.77 seconds
```

Dataset Processing & Dataset Enhancement

Filter conditions:

Passed all the testcases

&&

opt time/ total time <=0.5

Step 3:

Keep all the questions that were slow but correct in the previous step

M678

	A	B	C	D	E	F	G
1	problem	passed test	wrong answer	time limit exceed	total time	opt time	opt time / total time
372	1340	30	0	0	97.05	2.23	0.023
419	1522	30	0	0	32.96	1.36	0.041
434	5818	30	0	0	18.11	0.95	0.052
440	8907	30	0	0	18.37	0.99	0.054
503	5127	30	0	0	10.72	0.86	0.08
508	260	30	0	0	19.21	1.64	0.085
519	7843	30	0	0	10.81	0.92	0.085
520	4533	30	0	0	19.44	1.73	0.089
556	1020	30	0	0	16.73	1.85	0.111
562	2198	30	0	0	10.01	1.34	0.134
567	5133	30	0	0	6.87	0.92	0.134
573	3152	30	0	0	8.68	1.37	0.158
583	1980	30	0	0	6.59	1.3	0.197
599	8707	30	0	0	4.82	0.97	0.201
601	5481	30	0	0	4.22	0.99	0.235
604	10166	30	0	0	3.99	0.95	0.238
610	10527	30	0	0	3.58	0.97	0.271
619	35	30	0	0	3.6	1.24	0.344
623	9393	30	0	0	4.2	1.59	0.379
629	3751	30	0	0	3.2	1.29	0.403
632	6061	30	0	0	2.2	0.98	0.445
634	7328	30	0	0	2.44	1.09	0.447
635	1209	30	0	0	2.66	1.26	0.474
648	2690	30	0	0	2.92	1.41	0.483
654	1820	30	0	0	2.62	1.3	0.496
655	6536	30	0	0	2.05	1.02	0.498
659	12345	30	0	0	2.1	1.05	0.5

Dataset Processing & Dataset Enhancement

File structure

```
├── question.txt
├── canonical_solution.cpp
├── canonical_solution.java
├── canonical_solution.py
├── input_output.json
└── metadata.json
```

Dataset Processing & Dataset Enhancement

Statistical data of our dataset

Supported Language	Number of Problems
C++ only	52
Java only	18
Python only	32
Python and C++	1
Java and C++	7
C++, Java and Python	1
Total	111

Bechmark Creation

- Metrics
  - Total Time (TT)
  - Efficiency Level (EL)
  - Timeout Rate (TR)
  - Pass@1
  - Optimal solution ratio (Opt)
- Code Execution Framework

Bechmark Creation

Metrics

- Total Time (TT)

$$TT = \frac{1}{N} \sum t_{\text{gen}}$$

- Efficiency Level (EL)

$$G = \{G_1, G_2, \ldots, G_n\}$$

$$O = \{O_1, O_2, \ldots, O_n\}$$

$$EL_k = \frac{\sum_{O_i \in O} O_i}{\sum_{G_i \in G} G_i}$$

$$\%EL = \frac{1}{N} \sum_{K=1}^{N} EL_k * 100\%$$

Bechmark Creation

Metrics

- Timeout Rate (TR)
- Pass@1

$$\text{pass@1} := \mathop{\mathbb{E}}_{\text{Problems}}\left[1 - \frac{\binom{n-c}{1}}{\binom{n}{1}}\right]$$

- Optimal solution ratio (Opt)

$$\frac{t_{\text{gen}} - t_{\text{opt}}}{t_{\text{opt}}} < \theta$$

Bechmark Creation

Code Evaluation Framework

```
(base) canranliu@Canrans-MacBook-Pro timeEval % python test_print.py
Please enter the language you want to test (python, cpp, java):
```

	A	B	C	D	E	F	G	H	I	J
1	problem	passed tests	wrong answers	e limit excee	opt_time	TT	EL	TR	Pass@1	Opt
2	98	30	0	0	1.13	11.89	0.095	0	1	0
3	99	30	0	0	0.85	1.28	0.667	0	1	0
4	111	0	30	0	1.3	9.19	0	0	0	0
5	116	1	29	0	0.87	1.61	0.508	0	0	0
6	118	30	0	0	1.4	1.38	1	0	1	1
7	124	6	18	6	1.01	36.41	0.552	0.2	0	0
8	133	30	0	0	0.83	1.26	0.662	0	1	0
9	135	0	30	0	0.82	1.27	0	0	0	0
10	136	30	0	0	0.89	1.27	0.703	0	1	1
11	140	30	0	0	0.82	1.33	0.615	0	1	0
12	145	30	0	0	0.84	1.3	0.651	0	1	0
13	157	30	0	0	0.83	1.92	0.434	0	1	0
14	172	30	0	0	1.12	11.85	0.094	0	1	0
15	184	0	30	0	0.83	1.27	0	0	0	0
16	185	0	30	0	0.87	1.3	0	0	0	0

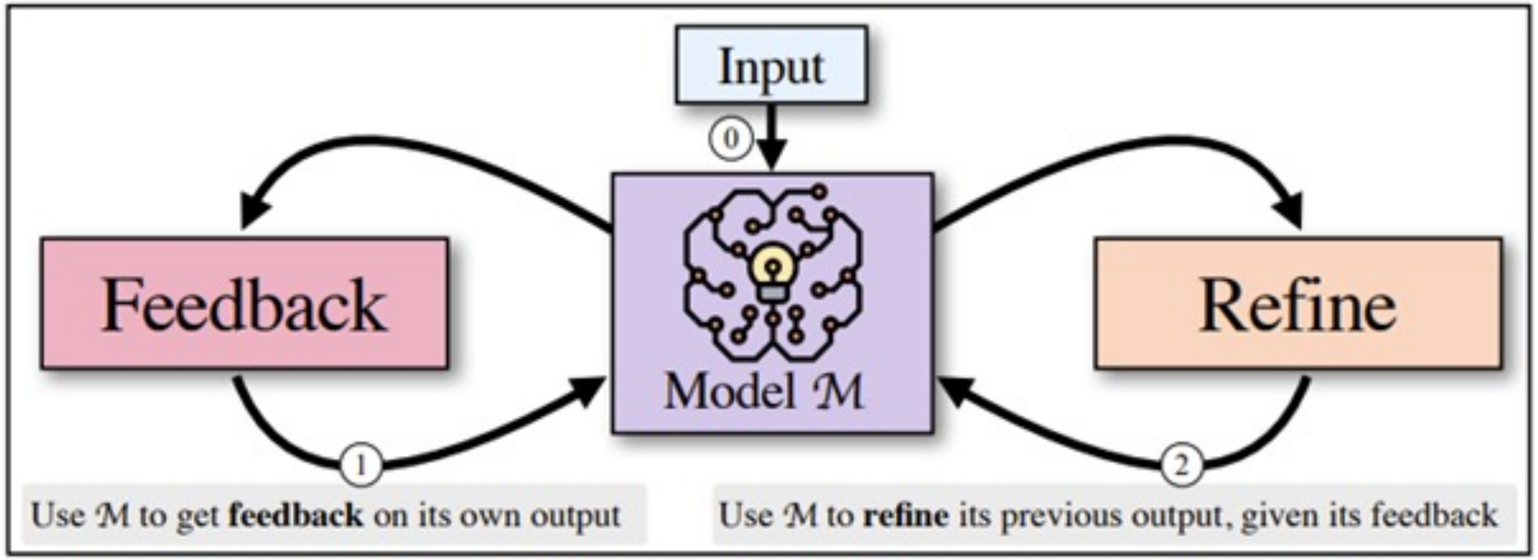
Contents

4. Empirical Study

- **Self-Refine**
- **Multi-Agent Collaboration**

Self-refine: Overview

- *"SELF-REFINE: Iterative Refinement with Self-Feedback"*



Self-refine: Process

- **Initialization Phase:**

  - The model is first provided with a correct yet slow version of code, and it is tasked to directly generate an optimized version of this code.

- **Feedback Phase:**

  - The optimized version of code is given back to the model to obtain feedback.

- **Refine Phase:**

  - Refine the code based on the feedback.

Self-refine: Prompt

- **Initialization Prompt (few-shot):**

 # slower version:
{Slow code}
 # optimized version of the same code:
{Optimized code}
END

More examples...

Few-shot examples

 # slower version:
{The correct but slow code provided by timeEval}
 # optimized version of the same code:

Self-refine: Prompt

- **Feedback Prompt (few-shot):**

{slow code}
 # Why is this code slow?
{feedback}
END

More examples...

Few-shot examples

{The correct but slow code provided by TimeEval}
 # Why is this code slow?

Self-refine: Prompt

- **Refine Prompt (zero-shot):**

{The correct but slow code provided by TimeEval}

 # Why is this code slow?

{Feedback from the model}

How to improve this code? Please provide the improved version of the code.

Self-refine: Result

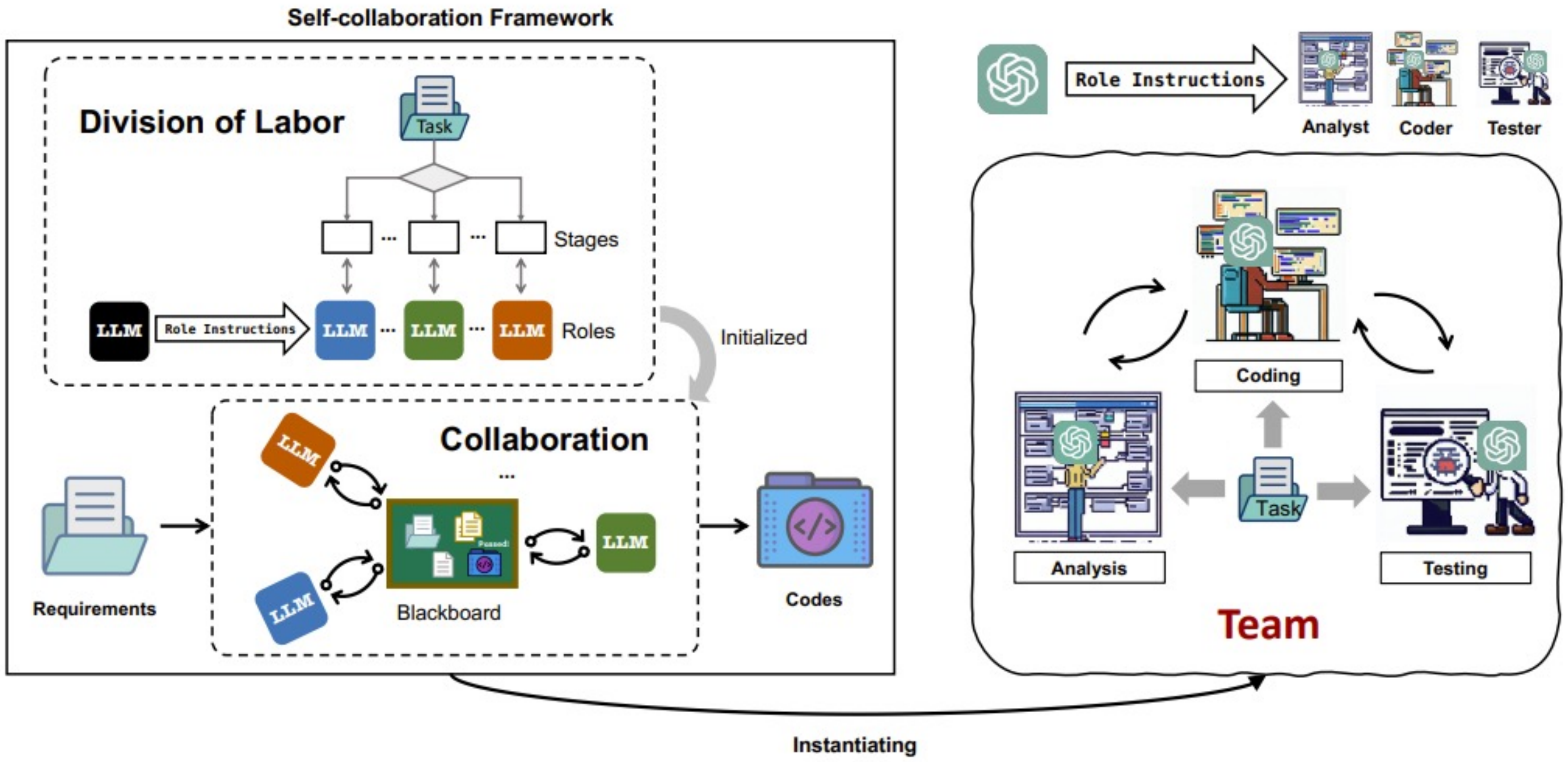
Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Self-refine	7.0	41.9	2.5	61.8	11.8
C++	**baseline**	3.5	40.4	0	100	16.4
	Self-refine	4.3	63.4	2.2	52.5	37.7
Java	**baseline**	12.6	24.0	0	100	3.8
	Self-refine	7.5	30.7	2.6	46.1	7.3

Self-refine: Case Study

- Wrong when initialization: 12/20
- Wrong when 1st round of self-refine: 5/20
- Wrong when 2nd round of self-refine: 1/20
- Wrong when 4th round of self-refine: 2/20

Multi-Agent Collaboration: Overview

- *"Self-collaboration Code Generation via ChatGPT"*



Multi-Agent Collaboration: Process

- **Analysis Phase:**

  - The task is first given to the Analyst, who then writes a **high-level plan** based on the task requirements.

- **Coding Phase:**

  - Then, this plan is passed on to the Coder, who writes the corresponding **code** according to the plan.

- **Testing and Iteration Phase:**

  - The completed code is handed over to the Tester for testing, and the Tester summarizes the test results into a **report**.

  - If the code passes the test, the process ends, and the correct code is output.

  - If the test fails, the test report is fed back to the Coder, who then tries to correct the code.

Multi-Agent Collaboration: Prompt

- Role Instruction

Role Instructions = Team Description + User Requirment + Role Description	
Team Description	There is a development team that includes a requirements analyst, a developer, and a quality assurance tester. The team needs to develop programs that satisfy the requirements of the users. The different roles have different divisions of labor and need to cooperate with each others.
User Requirment	The requirement from users is '{Requirment}'. *For example: {Requirment} = Input to this function is a string containing multiple groups of nested parentheses. Your goal is to separate those group into separate strings and return the list of those. Separate groups are balanced (each open brace is properly closed) and not nested within each other Ignore any spaces in the input string*
Role Description	**Coder:** I want you to act as a developer on our development team. You will receive plans from a requirements analyst or test reports from a tester. Your job is split into two parts: 1. If you receive a plan from a requirements analyst, write code in Python that meets the requirements following the plan. Ensure that the code you write is efficient, readable, and follows best practices. 2. If you receive a test report from a tester, fix or improve the code based on the content of the report. Ensure that any changes made to the code do not introduce new bugs or negatively impact the performance of the code. Remember, do not need to explain the code you wrote.

Multi-Agent Collaboration: Result

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Multi-Agent Collaboration	24.6	53.0	17.0	20.1	5.9
C++	**baseline**	3.5	40.4	0	100	16.4
	Multi-Agent Collaboration	11.9	39.8	6.0	55.7	16.4
Java	**baseline**	12.6	24.0	0	100	3.8
	Multi-Agent Collaboration	16.0	39.0	6.7	57.7	3.8

Multi-Agent Collaboration - Adjust

- Prompt of Tester:

Tester = team description + user requirement +

“I want you to act as a quality assurance tester on our development team. You will receive code from a developer. Your job is:

1. Test the functionality of the code to ensure it satisfies the requirements.
2. **Test the efficiency of the code to ensure it has good time complexity.**
3. Write reports on any issues or bugs you encounter.
4. If the code or the revised code has passed your tests, write a conclusion ’Code Test Passed’.

Remember, the report should be as concise as possible, without sacrificing clarity and completeness of information. Do not include any error handling or exception handling suggestions in your report.” + “The code from a developer is: {script}”.

Multi-Agent Collaboration: Result

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Multi-Agent Collaboration	24.6	53.0	17.0	20.1	5.9
	Multi-agent collaboration with new Tester	21.8	46.7	14.3	26.5	5.9
C++	**baseline**	3.5	40.4	0	100	16.4
	Multi-Agent Collaboration	11.9	39.8	6.0	55.7	16.4
	Multi-agent collaboration with new Tester	8.3	39.0	4.1	50.8	18.0
Java	**baseline**	12.6	24.0	0	100	3.8
	Multi-Agent Collaboration	16.0	39.0	6.7	57.7	3.8
	Multi-agent collaboration with new Tester	16.0	39.0	6.7	57.7	3.8

Multi-Agent Collaboration: Case Study

Type	Number
Correct but low-efficient plan, timeout code, and useless tester	6
Correct but low-efficient plan, wrong code, and useless tester	11
Wrong plan, wrong code, and useless tester	1
Others	2

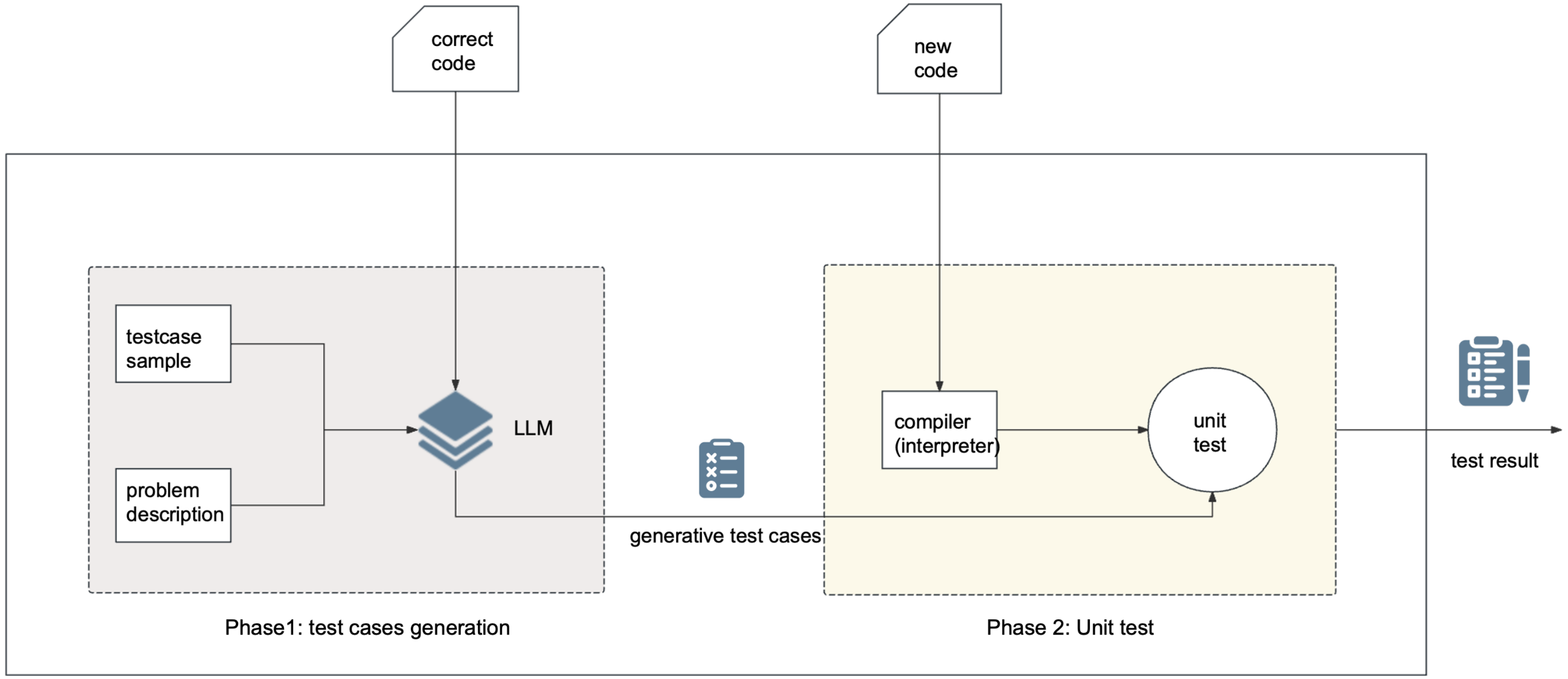
Contents

5. Methodology

- Generative Executor Module
- Self-Refine-Executor Framework
- Multi-Agent-Executor Framework

Generative Executor Module

Generative Executor



Generative Executor Module

Phase 1: Testcases generation

```
(base) canranliu@Canrans-MacBook-Pro UT % python main.py
Generating testcases based on correct code...
  0%|                                                                      | 0/1 [00:00<?, ?it/s]
Generating testcases for problem: ./098/question.txt
100%|██████████████████████████████████████████████████████████████████| 1/1 [00:01<00:00,  1.67s/it]
Input testcases: ['3 4\n0110\n1010\n0111\n', '2 3\n101\n010\n', '4 5\n11111\n00000\n11111\n00000\n', '1 1\n1\n']
Output testcases: ['2\n', '0\n', '0\n', '0\n']
```

Generative Executor Module

Phase 2: Unit test and feedback generation

```
Fail
An error occurred in the program:
./tmp/Main.java:23: error: not a statement
                for (int k = 0; k < n; k++) {adf
                                             ^
./tmp/Main.java:23: error: ';' expected
                for (int k = 0; k < n; k++) {adf
                                                ^
2 errors
```

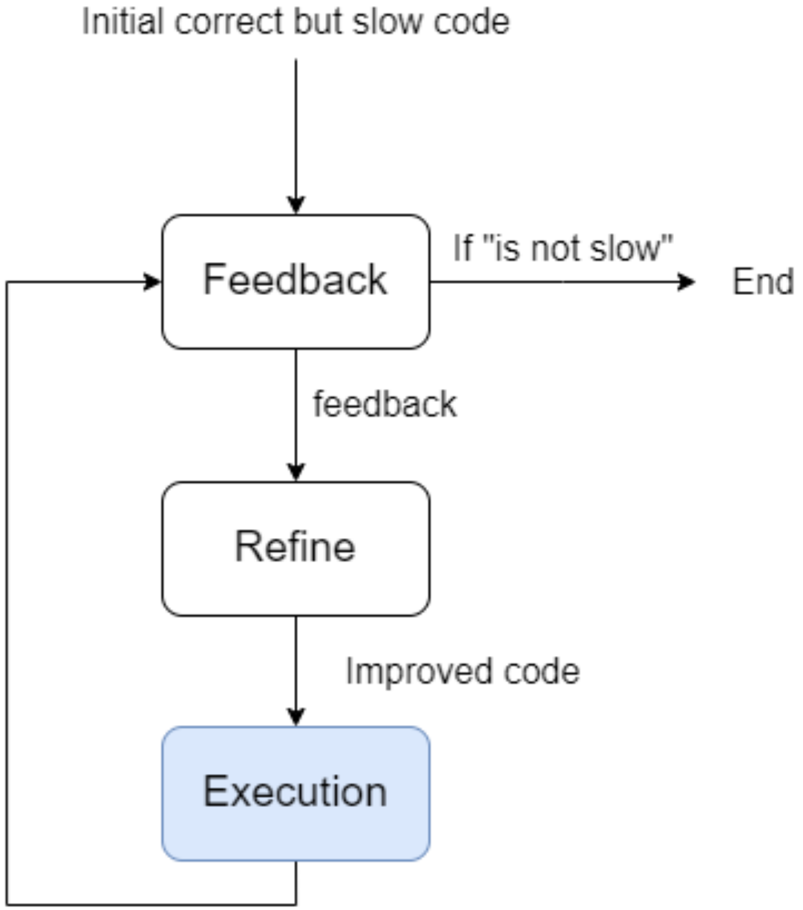
```
Fail
The new code failed following testcases:
When the input is 3 4
0110
1010
0111
The expected output is 2
The output of the new code is -1

When the input is 2 3
101
010
The expected output is 0
The output of the new code is -2

When the input is 4 5
11111
00000
11111
00000
The expected output is 0
The output of the new code is -10
```

Self-Refine-Executor Framework - Motivation

- The feedback phase only focuses on code efficiency, which often lead to errors in the refined code.

- The subsequent self-refinements cannot correct the errors, leading to the worse code.

Self-Refine-Executor Framework - Design



Self-Refine-Executor Framework - Design

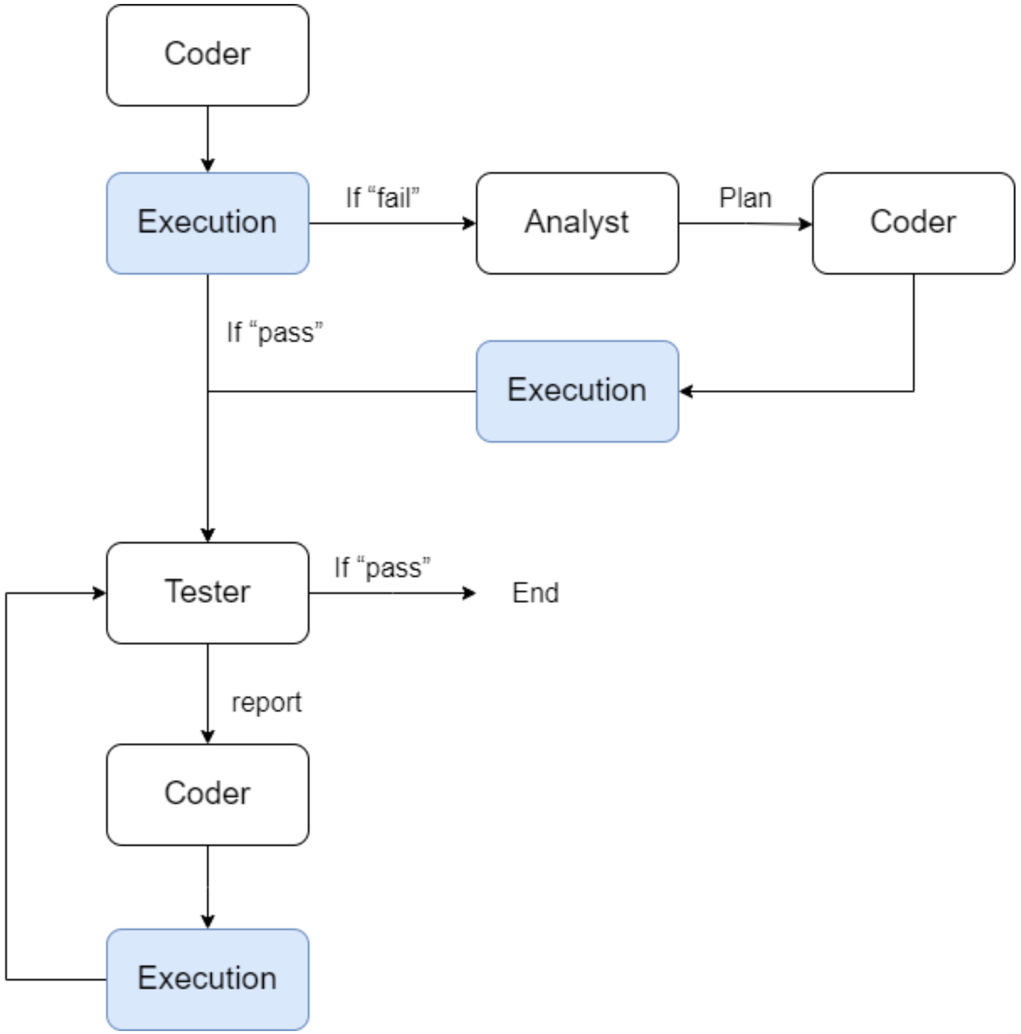
- **Initialization Phase:**
  - The model is first provided with a correct but slow version of code, and it is tasked to directly generate an optimized version of this code.
- **Execution Phase:**
  - Submit the code for testing by the execution module.
  - If the test result is “pass”, the code is retained.
  - If it fails, the code is discarded, and the previous correct code is used for the next feedback and refinement.
- **Feedback Phase:**
  - The optimized version of code is given back to the model to obtain feedback.
- **Refine Phase:**
  - Refine the code based on the feedback.

Multi-Agent-Executor Framework - Motivation

- The plans given by the Analyst are generally correct but often inefficient;

- The Tester is not able to effectively detect obvious errors and judge the efficiency of the code.

Multi-Agent-Executor Framework - Design



Multi-Agent-Executor Framework - Design

- **Initialization Phase:**
  - The task is firstly given to the Coder, who will write code according to the requirements of users.
  - The code will then be passed to the Executing Phase directly.
  - If the execution result of this initial code is “Pass”, it then goes to the Testing Phase.
  - If the code fails, the Analyst would be called to give a high-level plan for this task.

- **Coding Phase:**

  - This plan is passed back to the Coder, and then the Coder will write the code according to the plan.

Multi-Agent-Executor Framework - Design

- **Executing Phase:**
  - The code will be executed through the external “Generative Executor module”.
  - The module returns a result, indicating “Pass” if the code passes all test cases, or “Fail” along with the test cases that failed and any error information (if available).
- **Testing and Iteration Phase:**
  - The execution result is given to the Tester.
  - If the result is “Pass”, the Tester analyzes whether there is room to improve the efficiency of the code;
  - If the result is “Fail”, the Tester drafts a report based on the error information.
  - If the code is correct and the Tester believes it is efficient enough, the iteration ends.
- **Repairing Phase:**
  - If the test is not passed, the test report is sent back to the Coder, who revises the code according to the report.

Contents

6. Experiment

- Baseline Experiment
- Self-Refine-Executor
- Multi-Agent-Executor
- In-context Learning
- Others

Baseline

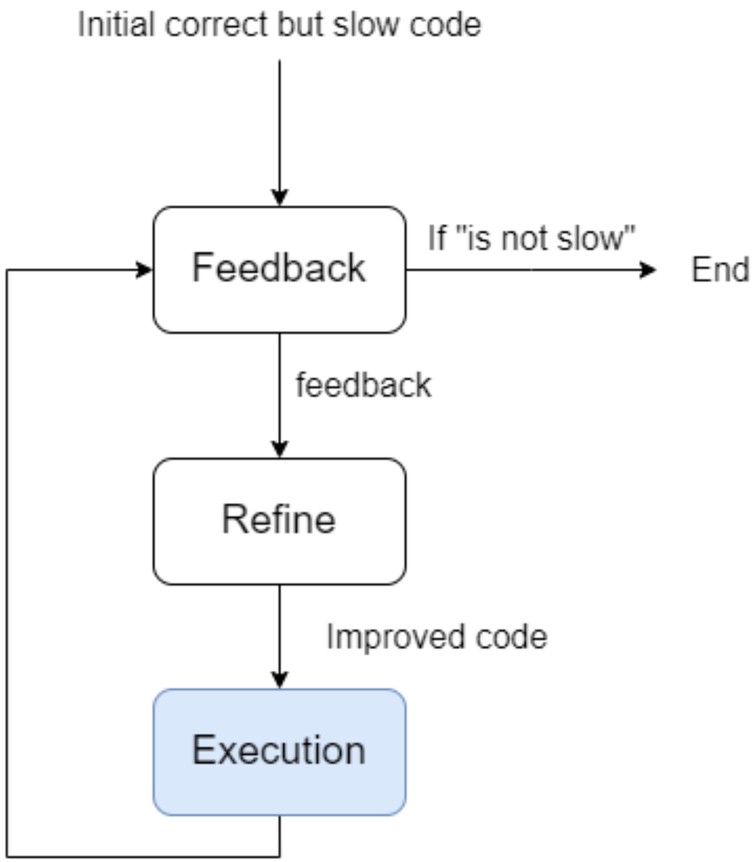
- Prompt:
  - Please generate *{language}* code that can be run directly to solve the following programming problem. Do not add any text description!

Baseline

- Prompt:
  - Please generate *{language}* code that can be run directly to solve the following programming problem. Do not add any text description!
- Result:

Language	Experiment	Total Time(TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
CPP	**baseline**	3.5	40.4	0	100	16.4
Java	**baseline**	12.6	24.0	0	100	3.8

Self-Refine-Executor



Self-Refine-Executor: Result

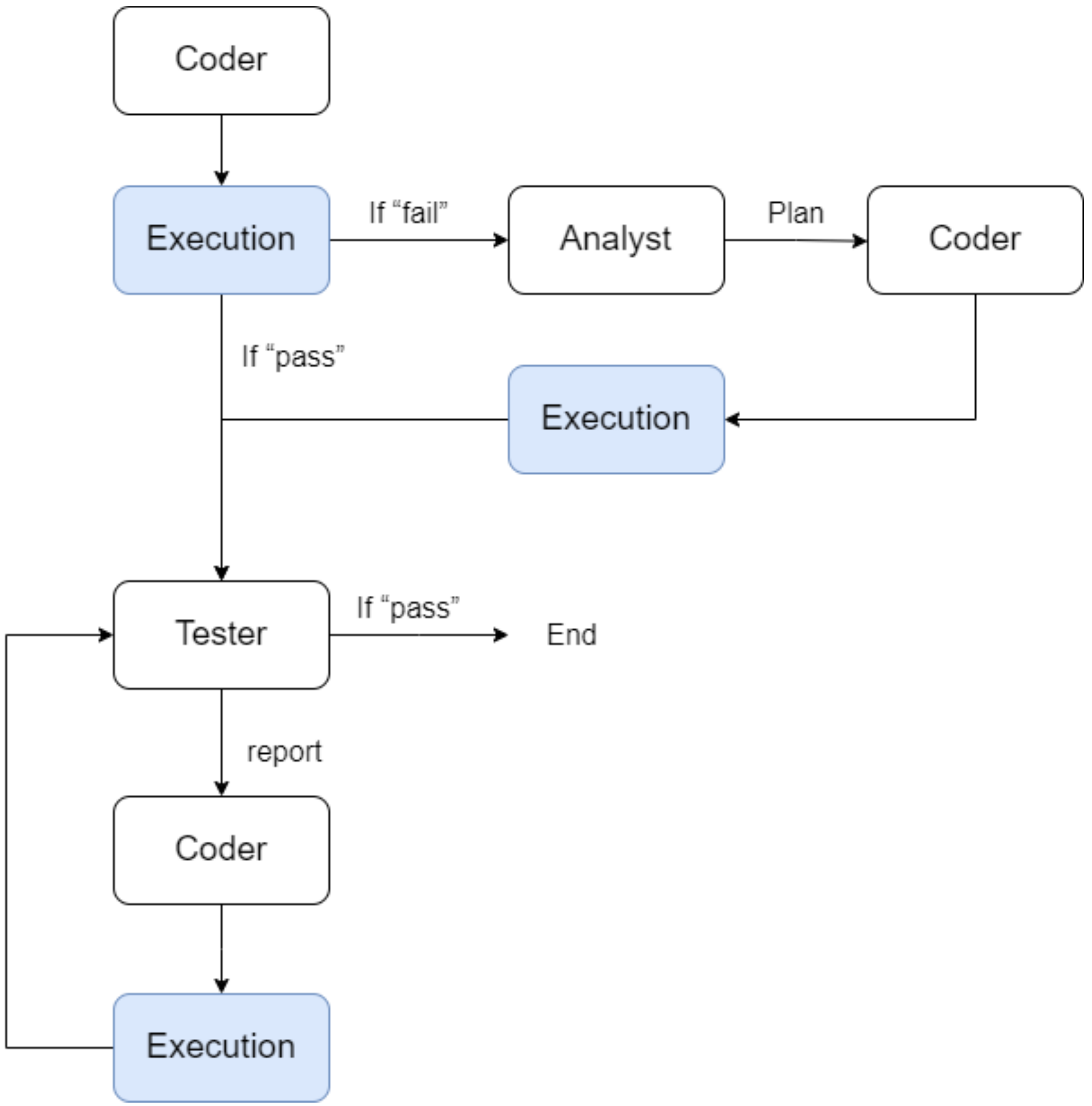
Language	Experiment	Total Time(TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Self-refine	7.0	41.9	2.5	61.8	11.8
	Self-refine-executor	7.1	40.2	2.3	91.2	17.6
CPP	**baseline**	3.5	40.4	0	100	16.4
	Self-refine	4.3	63.4	2.2	52.5	37.7
	Self-refine-executor	3.4	53.8	0.7	90.1	29.5
Java	**baseline**	12.6	24.0	0	100	3.8
	Self-refine	7.5	30.7	2.6	46.1	7.3
	Self-refine-executor	7.6	33.2	0.9	87.3	4.4

Self-Refine-Executor: Result

Language	Experiment	Total Time(TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Self-refine	7.0	41.9	2.5	61.8	11.8
	Self-refine-executor	7.1	40.2	2.3	91.2	17.6
CPP	**baseline**	3.5	40.4	0	100	16.4
	Self-refine	4.3	63.4	2.2	52.5	37.7
	Self-refine-executor	3.4	53.8	0.7	90.1	29.5
Java	**baseline**	12.6	24.0	0	100	3.8
	Self-refine	7.5	30.7	2.6	46.1	7.3
	Self-refine-executor	7.6	33.2	0.9	87.3	4.4

- Why is the pass@1 not 100%?
  - After self-refine, the efficiency of the code actually decreased, but the executor-generated test cases were not large enough to detect timeout situations.
  - After self-refine, the optimized code had errors, but the executor-generated test cases were not comprehensive enough to detect these errors.

Multi-Agent-Executor

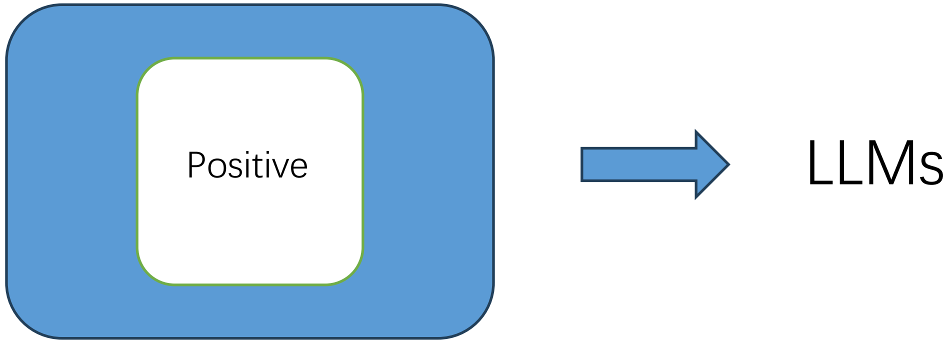


Multi-Agent-Executor

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Multi-Agent Collaboration	24.6	53.0	17.0	20.1	5.9
	Multi-agent collaboration with new Tester	21.8	46.7	14.3	26.5	5.9
	Multi-Agent-Executor	10.2	53.2	4.6	73.5	14.7
C++	**baseline**	3.5	40.4	0	100	16.4
	Multi-Agent Collaboration	11.9	39.8	6.0	55.7	16.4
	Multi-agent collaboration with new Tester	8.3	39.0	4.1	50.8	18.0
	Multi-Agent-Executor	8.9	63.7	3.4	70.2	32.8
Java	**baseline**	12.6	24.0	0	100	3.8
	Multi-Agent Collaboration	16.0	39.0	6.7	57.7	3.8
	Multi-agent collaboration with new Tester	16.0	39.0	6.7	57.7	3.8
	Multi-Agent-Executor	15.8	45.5	8.0	59.1	7.4

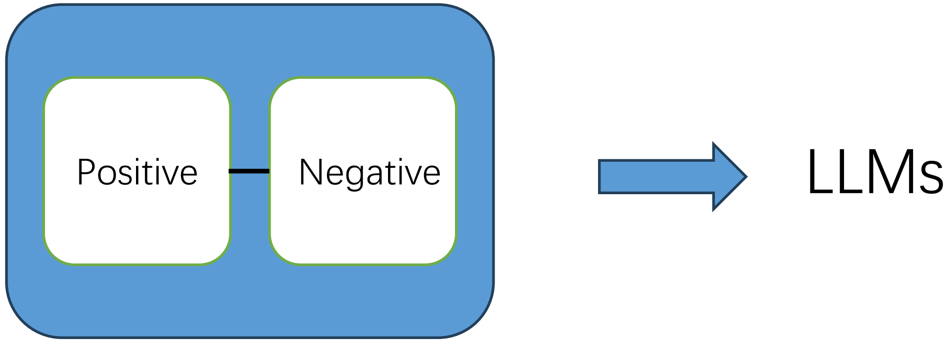
Experiment

Question + Positive Example



In-Context-Learning

Question + Positive Example + Negative Example



Experiment

In-Context-Learning

Problem Type	Negative	Positive
Binary search	$O(m+n)$	$O(\log(m+n))$
Divide and conquer	$O(n^2)$	$O(n)$
Dynamic programming	$O(n^3)$	$O(n)$
Sorting	$O(n \log n)$	$O(n)$

Table 18: Time Complexity of Different Problem Types

Experiment

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	baseline	6.0	31.5	0	100	8.8
	ICL (1 positive example)	7.4	32.1	6.8	26.5	2.9
	ICL (2 positive example)	3.1	36.4	2.0	50	8.8
	ICL (4 positive example)	3.4	32.5	2.5	50.0	8.8
	ICL (1 positive and negative example)	3.3	30.1	2.5	50.0	11.7
	ICL (2 positive and negative example)	3.7	39.3	2.0	58.8	8.8
	ICL (4 positive and negative example)	6.2	34.3	4.3	50.0	5.9

Experiment

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Java	baseline	12.6	24.0	0	100	3.8
	ICL (1 positive example)	10.3	29.8	2.6	65.4	0
	ICL (2 positive example)	8.8	28.4	1.5	73.0	0
	ICL (4 positive example)	7.3	23.9	0.5	69.2	3.8
	ICL (1 positive and negative example)	9.6	27.4	2.0	65.3	3.8
	ICL (2 positive and negative example)	9.2	26.4	1.5	69.2	0
	ICL (4 positive and negative example)	7.0	26.2	0	76.9	0

Experiment

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
C++	baseline	3.5	40.4	0	100	16.4
	ICL (1 positive example)	5.6	38.6	1.1	81.8	21.2
	ICL (2 positive example)	5.2	57.1	1.3	90.2	42.6
	ICL (4 positive example)	6.7	50.8	1.9	83.6	39.3
	ICL (1 positive and negative example)	5.4	48.9	1.3	85.2	26.2
	ICL (2 positive and negative example)	5.3	49.1	1.3	83.6	31.1
	ICL (4 positive and negative example)	5.6	48.5	1.4	80.3	23.0

Experiment

Change Prompt

```
def get_messages(prompt, language):
    messages = []
    system_prompt = "Please generate " + language + "code that
       can be run directly to solve the following programming
       problem. Do not add any text description!" + "Please pay
        attention to the time complexity of your solution."
    messages.append(
        {"role": "system", "content": system_prompt}
    )
    messages.append(
        {"role": "user", "content": prompt}
    )

    return messages
```

Experiment

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	Change Prompt	7.49	42.2	5.2	67.7	11.7
C++	**baseline**	3.5	40.4	0	100	16.4
	Change Prompt	4.8	50.1	0.6	80.3	37.7
Java	**baseline**	12.6	24.0	0	100	3.8
	Change Prompt	11.0	31.1	2.4	80.7	7.7

Experiment

Chain of Thought (CoT)

```
def get_messages(prompt, language):
    messages = []
    system_prompt = "Please generate " + language + "code to
       solve the following programming problem. Let's think it
       step by step."
    messages.append(
        {"role": "system", "content": system_prompt}
    )
    messages.append(
        {"role": "user", "content": prompt}
    )

    return messages
```

Experiment

Language	Experiment	Total Time (TT)	Efficiency Level (EL)	Timeout Rate (TR)	pass@1	%opt (Optimality)
Python	**baseline**	6.0	31.5	0	100	8.8
	CoT	14.6	40.5	11.3	50.0	2.9
C++	**baseline**	3.5	40.4	0	100	16.4
	CoT	3.2	54.9	0.7	77.0	31.1
Java	**baseline**	12.6	24.0	0	100	3.8
	CoT	9.6	39.8	2.0	69.2	3.8

Conclusion

- Measure and process code contests dataset.
- We have improved the timeEval benchmark.
- We did the empirical study of the existing method.
- We proposed several frameworks and finally achieved satisfactory results.

Q&A Session

Thank you