LLM-Based Agents for Code Generation: Agent Coder [1]

University of Trieste, July 2024

Students:

Agustin Campagnolo

Simone Cappiello

DATA SCIENCE & SCIENTIFIC COMPUTING

1 https://doi.org/10.48550/arXiv.2312.13010

AGENTIC REASONING DESIGN PATTERNS

Reflection	→	self-feedback mechanism to reduce hallucinations
Tool Use	→	ability to use external resources or tools to achieve agent's goals
Planning	→	generation of a sequence of actions to achieve agent's goals
Multi-Agent Collaboration	→	multiple agents coordinating their actions to achieve a common goal

AGENT CODER
MULTI-AGENT FRAMEWORK

Programmer: generates and refines code based on feedback from test executor.

Test Designer: generates effective test cases for the generated code.

Test Executor: runs the code with the test cases and reports output and eventual errors to the programmer.

BENCHMARKING DATASETS:
HumanEval

→ developed by OpenAI and released in 2021, it has become one of the most widely used benchmarks for assessing code generation accuracy.

→ 164 hand-crafted programming challenges, each including a function signature, docstring, body, and unit tests

BENCHMARKING DATASETS:
HumanEval Example

```python
def cycpattern_check(a , b):
    """You are given 2 words. You need to return True
    if the second word or any of its rotations is a substring in the first word
    cycpattern_check("abcd","abd") => False
    cycpattern_check("hello","ell") => True
    cycpattern_check("whassup","psus") => False
    cycpattern_check("abab","baa") => True
    cycpattern_check("efef","eeff") => False
    cycpattern_check("himenss","simen") => True
    """

# test
def check(candidate):
    assert  candidate("xyzw","xyw") == False , "test #0"
    assert  candidate("yello","ell") == True , "test #1"
    assert  candidate("whattup","ptut") == False , "test #2"
    assert  candidate("efef","fee") == True , "test #3"
    assert  candidate("abab","aabb") == False , "test #4"
    assert  candidate("winemtt","tinem") == True , "test #5"
```

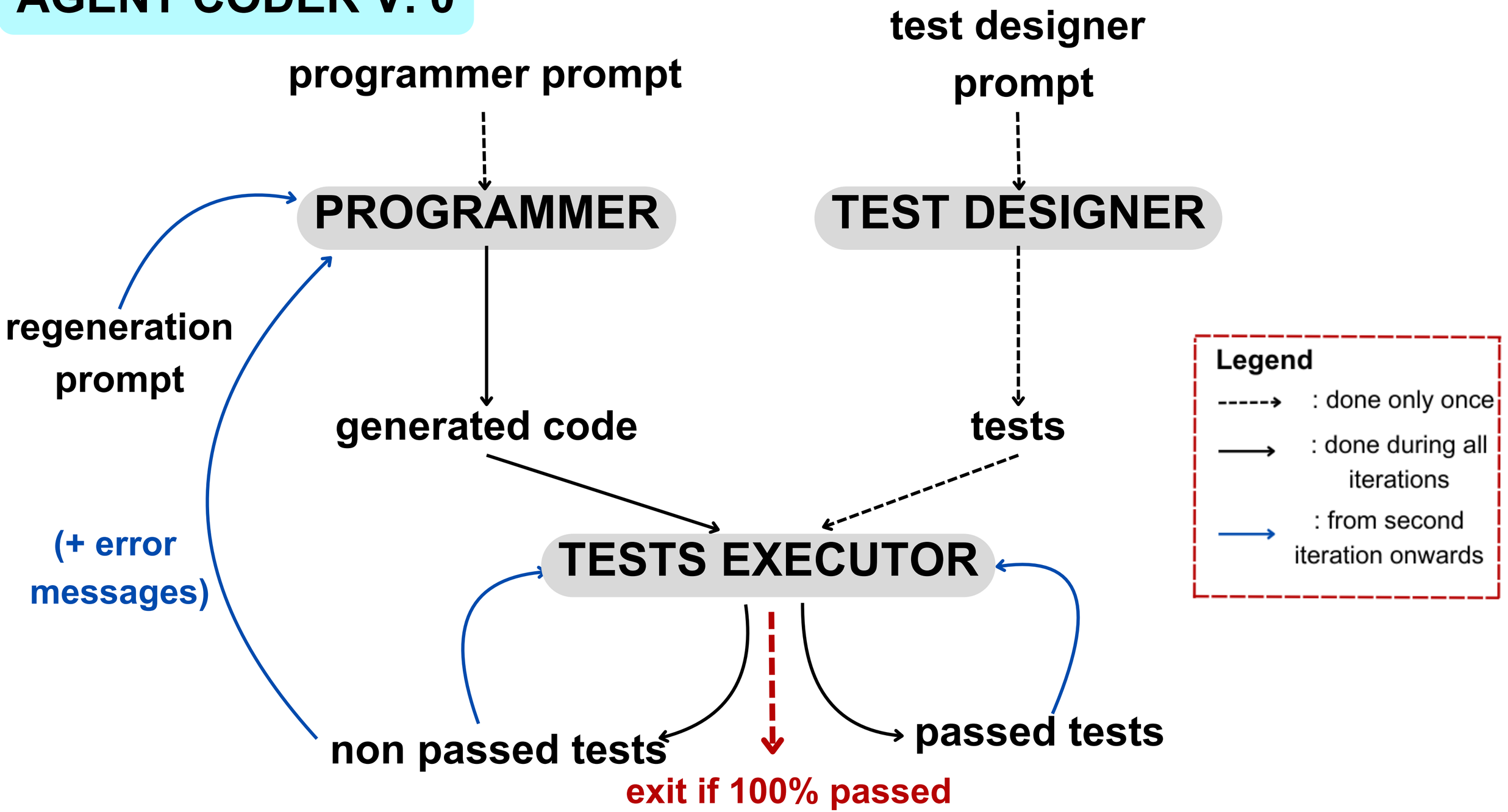
pass@k metric

Assesses model performance, considering a model successful w.r.t. a prompt if any of its 'k' generated solutions pass all tests.

$$\text{pass@k} = \frac{1}{N} \sum_{i=1}^{N} \left(1 - \prod_{j=1}^{k} (1 - c_{ij})\right) \in [0, 1]$$

c_{ij} : equals 1 if solution j is correct for problem i,
equals 0 otherwise

AGENT CODER V. 0

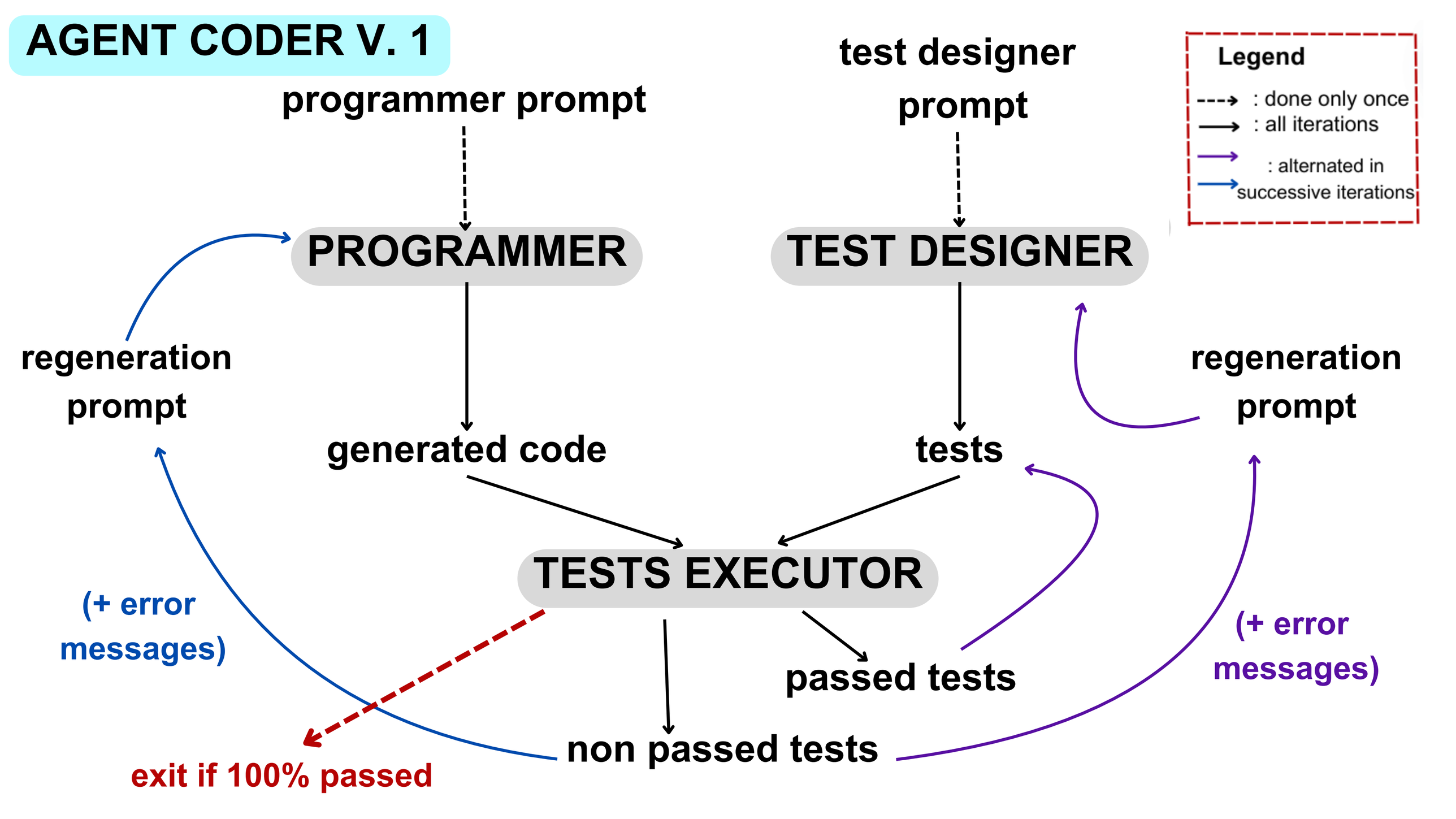


AGENT CODER (VERSION 0)

- Generation of tests happens only once, so the tests are fixed throughout the optimization loop.

- Accuracy of the generated test (computing using the canonical solution of the tasks present in the dataset):

Model	it. 0-2
GPT 4o-mini	85.19%


AGENT CODER V. 1
programmer prompt
test designer prompt
Legend
: done only once
: all iterations
: alternated in successive iterations
PROGRAMMER
TEST DESIGNER
regeneration prompt
regeneration prompt
generated code
tests
TESTS EXECUTOR
(+ error messages)
(+ error messages)
passed tests
non passed tests
exit if 100% passed

AGENT CODER (VERSION 1)

- Tests are checked and regenerated if wrong during the optimization procedure.

- Accuracy of the generated tests per iteration:

Model	it. 0	it. 1	it. 2	it. 3
GPT 4o-mini	85.19%	87.41%	87.78%	87.78%

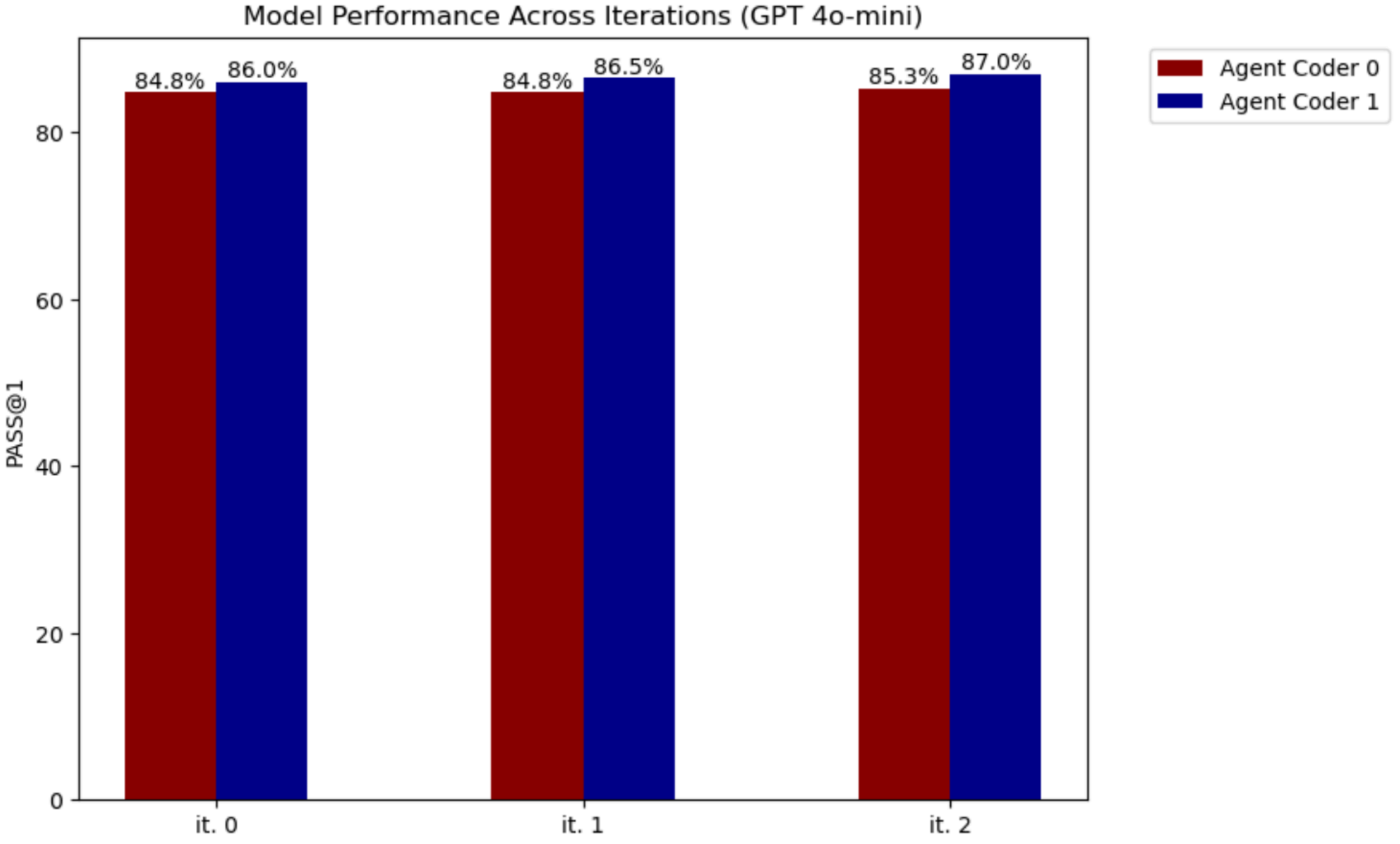
RESULTS COMPARISON
metric: PASS@1

On the whole dataset:

Version (model:GPT 4o-mini)	it. 0	it. 1	it. 2	Total tokens
Agent Coder 0	84.8%	84.8%	85.3%	**523544**
Agent Coder 1	86%	86.5%	**87%**	579602

Optimization in the long run (data from one chunk of the dataset):

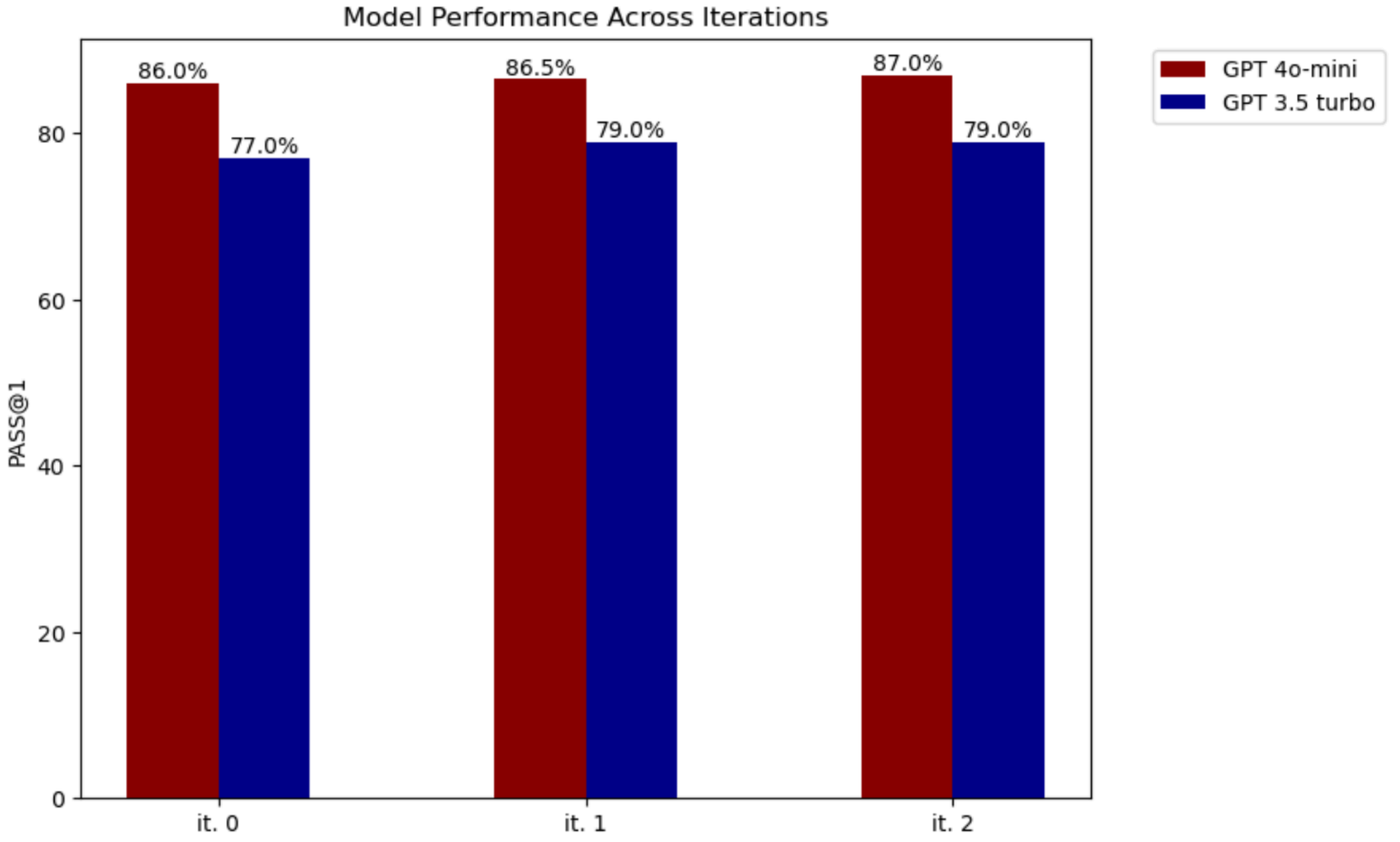
Model : GPT 4o-mini	it. 0	it. 1	it. 2	it. 3	it. 4	it. 5	it. 6
Agent Coder 0	79.63%	75.93%	81.48%	79.63%	75.93%	83.33%	75.93%
Agent Coder 1	81.48%	77.78%	79.63%	79.63%	81.48%	85.19%	**85.19%**


Model Performance Across Iterations (GPT 4o-mini)
PASS@1
0
20
40
60
80
84.8%
86.0%
84.8%
86.5%
85.3%
87.0%
it. 0
it. 1
it. 2
Agent Coder 0
Agent Coder 1

COMPARING DIFFERENT MODELS

We also compared the performance of GPT 3.5 Turbo vs GPT 4o-mini, the new lightweight and cost efficient OpenAI model.
Here the results for Agent Coder 1 on the whole HumanEval dataset:

Model	it. 0	it. 1	it. 2	Total tokens
GPT 4o-mini	86%	86.5%	**87%**	579602
GPT 3.5 turbo	77%	79%	79%	**572083**


Model Performance Across Iterations
PASS@1
0
20
40
60
80
86.0%
77.0%
86.5%
79.0%
87.0%
79.0%
it. 0
it. 1
it. 2
GPT 4o-mini
GPT 3.5 turbo

BENCHMARKING DATASETS:
MBPP

→ The MBPP (Mostly Basic Python Problems) benchmark is a dataset designed to evaluate the ability of machine learning models to understand and generate Python code.

→ It consists of 974 coding problems, each with a prompt, a solution, and test cases, intended to test models on code generation, execution, and understanding.

EXAMPLE MBPP

Prompt: “Write a python function to remove the first and last occurrence of a given character from the string.”

solution:

```python
def remove_Occ(s, ch):
    for i in range(len(s)):
        if s[i] == ch:
            s = s[0 : i] + s[i + 1:]
            break
    for i in range(len(s) - 1, -1, -1):
        if s[i] == ch:
            s = s[0 : i] + s[i + 1:]
            break
    return s
```

Tests:

```python
assert remove_Occ("hello", "l") == "heo"
assert remove_Occ("abcda", "a") == "bcd"
assert remove_Occ("PHP", "P") == "H"
```

RESULTS COMPARISON
metric: PASS@1

Tested on 55 entries of the dataset:

Version (model:GPT 4o-mini)	it. 0	it. 1	it. 2	it. 3	Total tokens
Agent Coder 0	75.47%	69.81%	69.81%	71.70%	**262191**
Agent Coder 1	73.58%	75.47%	75.47%	**77.36%**	284472

CHALLENGES AND LIMITATIONS

- No guarantee on their quality when comparing different models, though they do give information when comparing different optimization mechanisms.

- Standardized benchmarks such as HumanEval and MBPP datasets require outputs in highly specific formats, limiting flexibility and complicating the evaluation process.

FUTURE IMPROVEMENTS

→ Different frequencies for test regeneration could have been explored further.

→ Try to substantially increase the number of generated tests and/or the number of proposed solutions.

→ Adapt the prompts to real case scenarios.

End of the presentation,
thank you for the attention