ANALIZA PYNGUIN

Anca Alexandru-Iulian
Beșel Marin-Adrian
Ionescu Mihai-Leonard
Leventiu Adrian-George
Popescu Andrei-George

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

INTRODUCERE

Pynguin este un instrument conceput pentru a automatiza crearea testelor unitare în Python, folosind diverse strategii de generare precum căutarea genetică sau randomizată.

AGENDA

- Strategii Disponibile
- Dynamosa
- Parametrii
- Strategie LLM
- Comparatie Teste
- Costuri
- Bug cheie API
- Prompt Ineficient

STRATEGII DISPONIBILE

- **DYNAMOSA** - algoritm evolutiv multi-obiectiv dinamic (Panichella et al., TSE 2018).
- **MOSA** – many-objective sorting algorithm pentru acoperire ramificată (Panichella et al., ICST 2015).
- **MIO –** Many Independent Objective algorithm (Arcuri, SBSE 2017).
- **WHOLE_SUITE –** generare de suite complete de teste (Fraser & Arcuri, EvoSuite).
- **RANDOM –** generare aleatoare ghidată prin feedback (similar Randoop).
- **RANDOM_TEST_CASE_SEARCH –** căutare aleatoare pe cazuri individuale de test.
- **RANDOM_TEST_SUITE_SEARCH –** căutare aleatoare pe suite de teste.
- **LLM –** utilizează un model de limbaj mare (LLM) pentru generarea testelor (fără execuție sau aserții din Pynguin).

DYNAMOSA

Provocări cheie:

- Spațiu mare de căutare în generarea testelor.
- Acoperirea simultană a multor obiective (ramuri, condiții).
- Evitarea redundanței în explorarea codului.
- Obiective conflictuale sau greu de atins.

Beneficii principale:

- Selectează doar obiectivele neacoperite la fiecare pas.
- Reduce timpul de generare a testelor.
- Crește eficiența și direcția evoluției.
- Performant pe cod complex.

PARAMETRII

- **–project-path:** directorul rădăcină al proiectului care conține codul sursă ce va fi testat.
- **–tests-output-path:** calea unde Pynguin va salva fișierele de teste generate.
- **–maximum_test_execution_timeout:** timpul maxim (în secunde) permis pentru execuția fiecărei metode testate.
- **–minimum-coverage:** pragul minim de acoperire (%) pe care Pynguin trebuie să îl atingă înainte de a considera testele complete.
- **–algorithm:** definește strategia folosită de Pynguin pentru generarea testelor
- –**seed:** în Pynguin controlează generatorul de numere aleatorii folosit în procesul de generare a testelor. Setarea unei valori fixe permite reproducerea exactă a testelor generate.

STRATEGIE LLM

Provocări Cheie:

- Importuri/Tipuri date incorecte.
- Cod generat necurat, omisiuni.
- Lipsă seed (inconsistență).
- Corner case-uri irelevante, teste neterminate.

Beneficii Principale:

- Teste mult mai relevante contextual.
- Corner case-uri corelate cu logica funcțiilor.
- Suită de teste mai extinsă.

TESTE DYNAMOSA

```
def test_case_1():
    bool_0 = True
    bool_1 = True
    list_0 = [bool_0, bool_1, bool_1]
    int_0 = module_0.lastStoneWeight(list_0)
    assert int_0 == 1
```

Categoria de teste	Procentaj total
Teste relevante și utile	45%
Teste cu input valid dar redundant	25%
Teste irelevante sau eronate	30%

```
def test_case_3():
    int_0 = -1827
    bool_0 = True
    list_0 = [int_0, bool_0, bool_0, bool_0]
    int_1 = module_0.lastStoneWeight(list_0)
    assert int_1 == 1828
    list_1 = []
    int_2 = module_0.lastStoneWeight(list_1)
    assert int_2 == 0
    int_3 = module_0.lastStoneWeight(list_1)
    assert int_3 == 0
    int_4 = module_0.lastStoneWeight(list_1)
    assert int_4 == 0
```

TESTE LLM

```
@pytest.mark.parametrize("stones, expected", [
  ([2, 7, 4, 1, 8, 1], 1), # Example case: after all collisions, one stone of
weight 1 remains
  ([1], 1),          # Single stone: the weight of the stone itself
  ([3, 3], 0),         # Two stones of equal weight: both are destroyed
  ([10, 4, 2, 10], 2),    # After collisions, one stone of weight 2 remains
  ([5, 5, 5, 5], 0),     # All stones have the same weight and are destroyed
  ([9, 3, 2, 10], 0),    # After collisions, no stones remain
  ([1, 2, 3, 4, 5, 6, 7, 8, 9], 1), # Complex case with multiple collisions
  ([], 0),           # No stones: result is 0
])
def test_lastStoneWeight(stones, expected):
  assert lastStoneWeight(stones) == expected
```

COSTURI

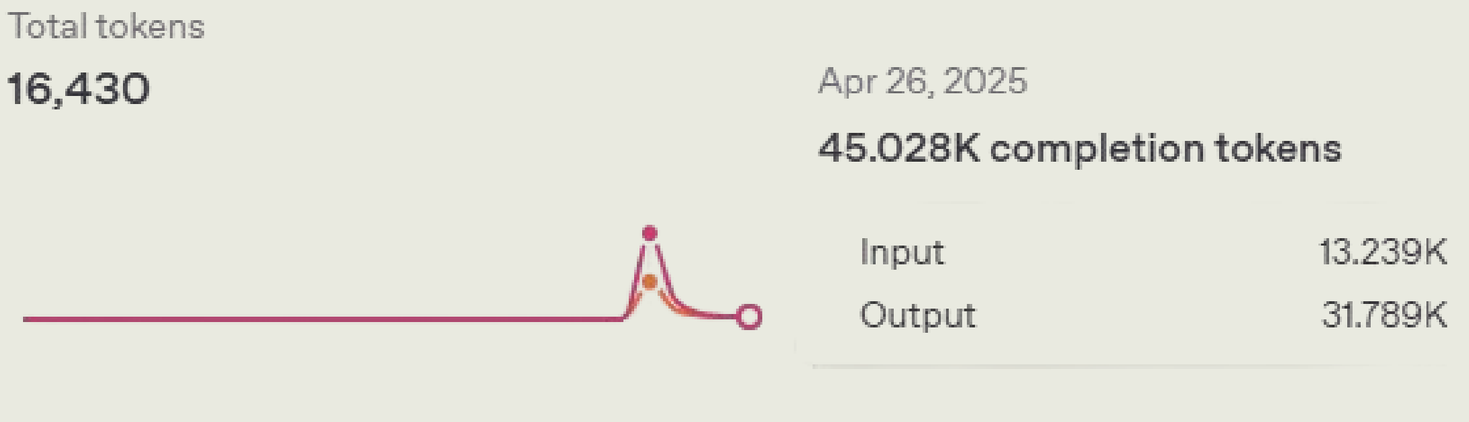
Costuri OpenAI calculate per milion de tokeni :

- Tokeni de intrare (input) : **2.50 USD**
- Tokeni de iesire (output) : **10.00 USD**

Consumul de tokeni in toate testele noastre a fost de :

- 16.432 tokeni de intrare
- 39.382 tokeni de iesire

Costul total platit pentru generarea testelor a fost de **0.42 USD**



BUG IDENTIFICAT

La momentul testării strategiei LLM din cadrul librăriei, am observat:

- **Problema identificată**: cheia de API, deși validă, era văzută de Pynguin ca fiind invalidă
- **Cauza inițială**: s-a suspectat migrarea cheilor de API de la OpenAI de la cele de useri, la cele de proiect
- **Cauza reală**: după mai multe investigații, am observat că problema provenea din conversia cheii la tipul de date **SecretStr**
- **Soluția**: înlocuirea acestei conversii

Am documentat un issue oficial pe GitHub: #93, pentru a ajuta și alți utilizatori și pentru a putea rezolva acel bug pentru toată lumea.

OpenAI API Key not working #93

Closed

leviaici opened yesterday

Describe the bug
Invalid API Key error whenever we are trying to use the LLM strategy and use a valid OpenAI API key (doesn't matter if user or project oriented API key).

To Reproduce
Steps to reproduce the behaviour:

1. Use Pynguin version '0.41.0dev'
2. Use whatever (minimal) code as a subject for test generation (the error is not related to the code).
3. Use the LLM strategy in the command line arguments.
4. Give the error (stack trace, etc) you are encountering: INVALID_API_KEY

Expected behavior
Accept the valid Open AI API Key.

Software Version (please complete the following information):

- OS: [Windows 11 Pro 24H2 (OS Build 26100.3775)]
- Python version [3.10.16]
- Pynguin Version [0.41.0dev]

Additional context
We found out that the issue comes with the SecretStr parsing method of the OPEN_AI_API_KEY. When we tried to remove the SecretStr, and directly pass it as a String, it worked.

☺ 👍 3

LuKrO2011 7 hours ago

I can confirm the issue and will provide a fix (that hopefully won't be removing SecretStr) ASAP.

☺

LuKrO2011 6 hours ago

The issue was resolved with `6f853ee`.
Thank you for reporting it!

☺

LuKrO2011 closed this as completed 6 hours ago

PROMPT INEFICIENT

```
# promptul original
user_prompt = "Generate test cases for the following Python code:\n\n"

# promptul modificat
user_prompt = "Generate test cases for the following Python code, import the necessary classes and function from the file named:" + config.configuration.module_name.replace(".", "/") +", be careful with the original code output type (if the output type is mentioned in the function, do not try to put something else instead of what it is required, make them so that the order of the result does not matter), the code:\n\n"
```

PROMPT INEFICIENT

Prompt original

```
import pytest
from your_module import kClosest

def test_kClosest_basic():
  points = [[1, 3], [-2, 2]]
  k = 1
  expected = [(-2, 2)]
  assert kClosest(points, k) == expected
```

Prompt modificat

```
import pytest
from cod import kClosest

def test_kClosest_all_points():
  points = [[1, 2], [2, 3], [3, 4]]
  k = 3
  result = kClosest(points, k)
  expected = [(1, 2), (2, 3), (3, 4)]
  assert set(result) == set(expected)
```

Mulțumim!