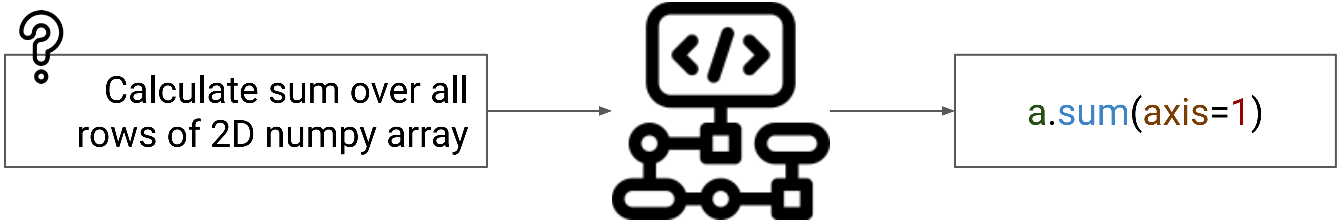
Natural language to Code Generation

Zora Wang, Nikitha Rao

Nov 9th, 2023

The NL2Code Task

Given a natural language instruction Q, generate code implementation C

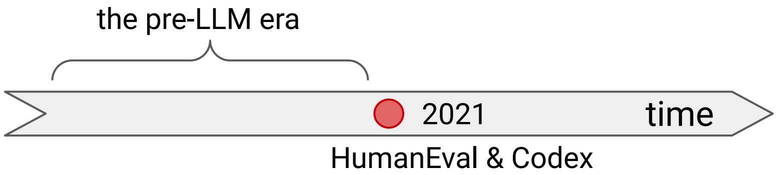


Calculate sum over all rows of 2D numpy array

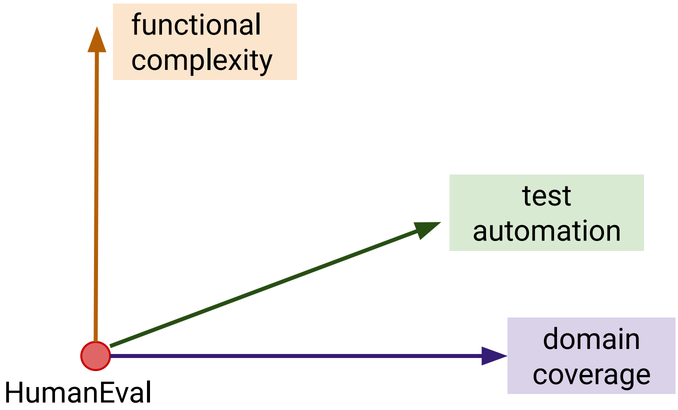
```
a.sum(axis=1)
```

The Landscape for NL2Code Generation

- Transition of Evaluation Metrics:
  - Lexical, neural based metrics
  - Test case execution



- Domain Coverage
  - Built-in grammar: “sum([1, 2, 4])
  - Domain-specific: data science
  - Open domain: diverse Python libraries
- Functional Complexity
  - Simple (toy) functions: e.g., LeetCode
  - Class level
  - Repository level
- Test Automation
  - Human-written tests
  - Fuzzing methods
  - Integrating LLMs



What We Had for Code Generation

Benchmark: **CodeXGLUE**, CoNaLa

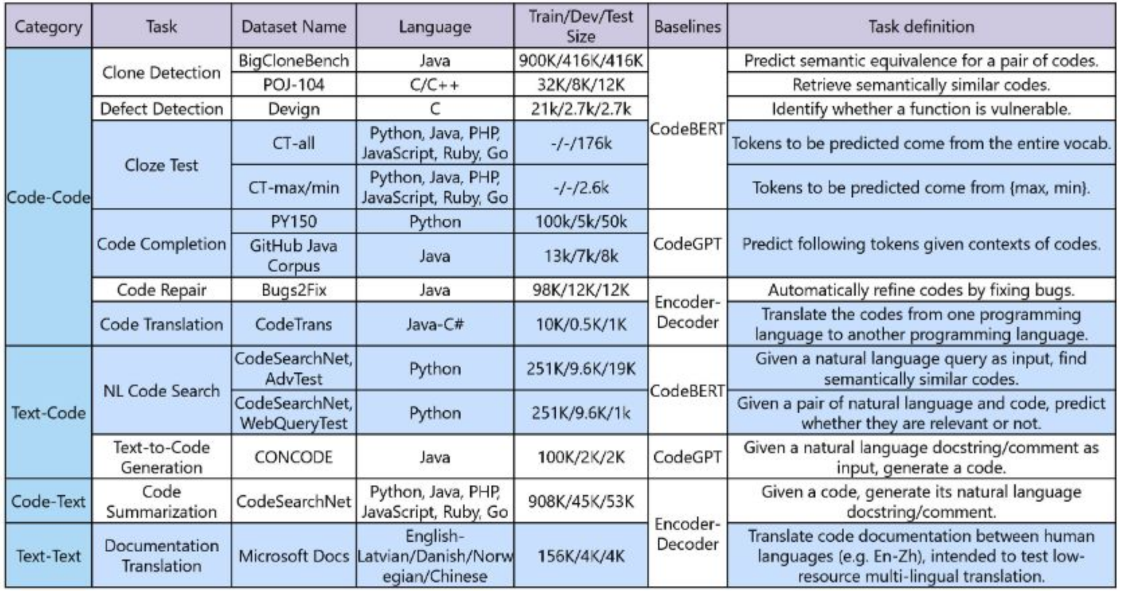
- code-code (clone detection, defect detection, cloze test, code completion, code repair, and code-to-code translation)
- text-code (natural language code search, text-to-code generation)
- code-text (code summarization)
- text-text (documentation translation)

Model

- CodeBERT, GraphCodeBERT

Evaluation Metrics

- Lexical based: BLEU
- Syntax based: CodeBLEU



Category	Task	Dataset Name	Language	Train/Dev/Test Size	Baselines	Task definition
Code-Code	Clone Detection	BigCloneBench	Java	900K/416K/416K	CodeBERT	Predict semantic equivalence for a pair of codes.
		POJ-104	C/C++	32K/8K/12K		Retrieve semantically similar codes.
	Defect Detection	Devign	C	21k/2.7k/2.7k		Identify whether a function is vulnerable.
	Cloze Test	CT-all	Python, Java, PHP, JavaScript, Ruby, Go	-/-/176k		Tokens to be predicted come from the entire vocab.
		CT-max/min	Python, Java, PHP, JavaScript, Ruby, Go	-/-/2.6k		Tokens to be predicted come from {max, min}.
	Code Completion	PY150	Python	100k/5k/50k	CodeGPT	Predict following tokens given contexts of codes.
		GitHub Java Corpus	Java	13k/7k/8k		
	Code Repair	Bugs2Fix	Java	98K/12K/12K	Encoder-Decoder	Automatically refine codes by fixing bugs.
	Code Translation	CodeTrans	Java-C#	10K/0.5K/1K		Translate the codes from one programming language to another programming language.
Text-Code	NL Code Search	CodeSearchNet, AdvTest	Python	251K/9.6K/19K	CodeBERT	Given a natural language query as input, find semantically similar codes.
		CodeSearchNet, WebQueryTest	Python	251K/9.6K/1k		Given a pair of natural language and code, predict whether they are relevant or not.
	Text-to-Code Generation	CONCODE	Java	100K/2K/2K	CodeGPT	Given a natural language docstring/comment as input, generate a code.
Code-Text	Code Summarization	CodeSearchNet	Python, Java, PHP, JavaScript, Ruby, Go	908K/45K/53K	Encoder-Decoder	Given a code, generate its natural language docstring/comment.
Text-Text	Documentation Translation	Microsoft Docs	English-Latvian/Danish/Norwegian/Chinese	156K/4K/4K		Translate code documentation between human languages (e.g. En-Zh), intended to test low-resource multi-lingual translation.

What We Had for Code Generation

Benchmark:

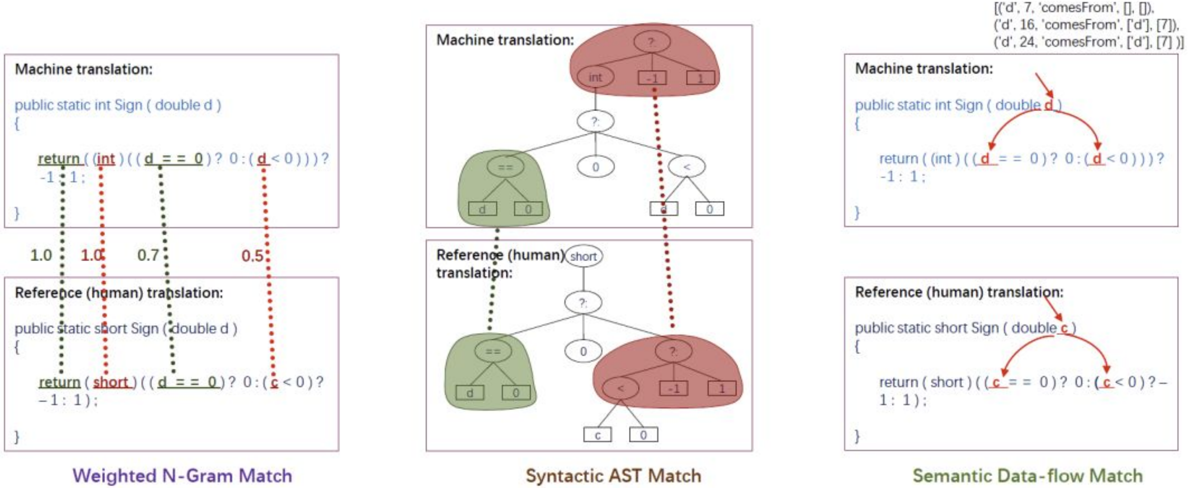
- CodeXGLUE, CoNaLa

Model

- CodeBERT, GraphCodeBERT

Evaluation Metrics

- Lexical based: BLEU
- Syntax based: **CodeBLEU**



Machine translation:

```
public static int Sign ( double d )
{
    return ( (int ) ( ( d  = =  0 ) ?  0 : ( d < 0 ) ) ) ?
    -1 : 1 ;
}
```

Reference (human) translation:

```
public static short Sign ( double d )
{
    return ( short ) ( ( d  = =  0 ) ?  0 : ( c < 0 ) ?
    –1 : 1 ) ;
}
```

Weighted N-Gram Match

Machine translation:

Reference (human) translation:

Syntactic AST Match

```
[('d', 7, 'comesFrom', [], []),
('d', 16, 'comesFrom', ['d'], [7]),
('d', 24, 'comesFrom', ['d'], [7] )]
```

Machine translation:

```
public static int Sign ( double d )
{
    return ( (int ) ( ( d  = =  0 ) ?  0 : ( d < 0 ) ) ) ?
    -1 : 1 ;
}
```

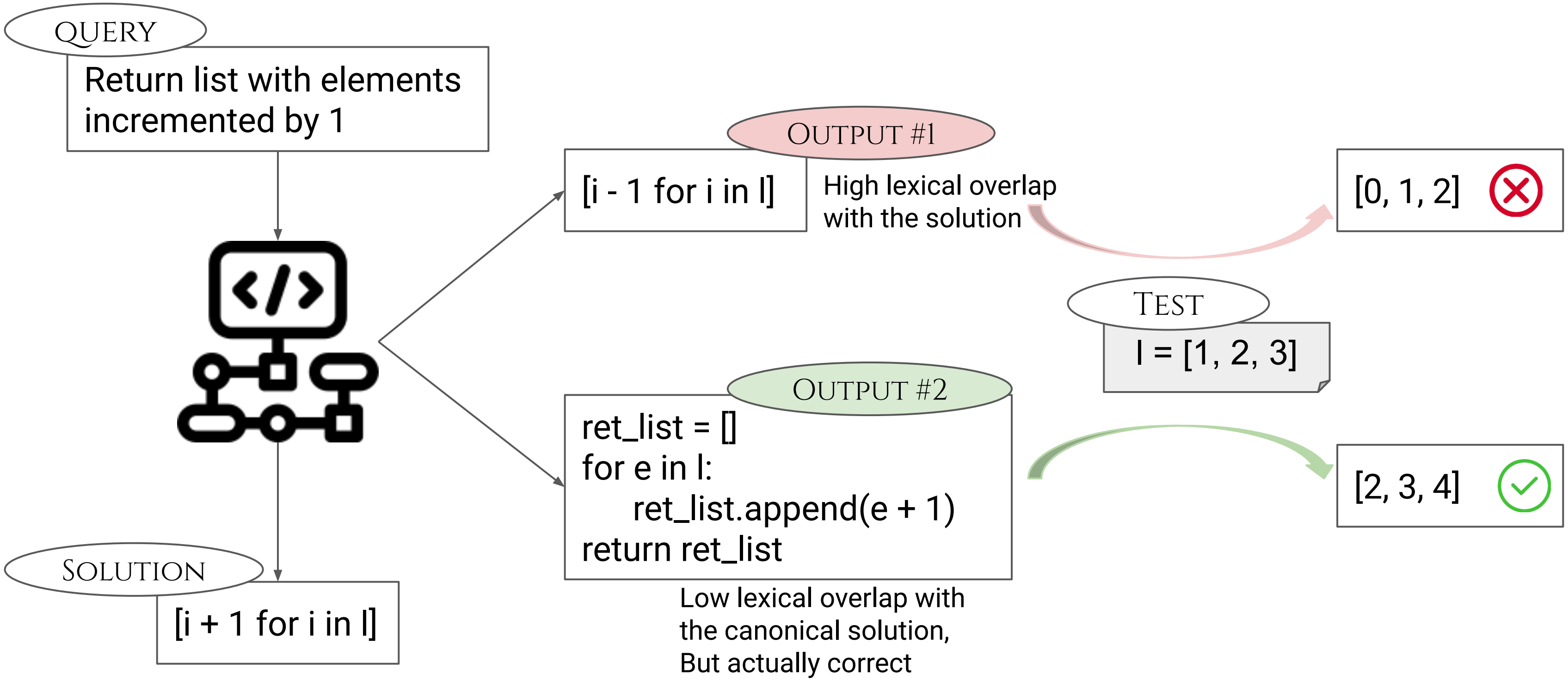
Reference (human) translation:

```
public static short Sign ( double c )
{
    return ( short ) ( ( c  = =  0 ) ?  0 : ( c < 0 ) ? –
    1 : 1 ) ;
}
```

Semantic Data-flow Match

CodeBLEU $= \alpha \cdot$ N − Gram Match (BLEU) $+ \beta \cdot$ Weighted N-Gram Match $+ \gamma \cdot$ Syntactic AST Match $+ \delta \cdot$ Semantic Data-flow Match

Issues: Evaluations Are Not Rigorous



QUERY

Return list with elements incremented by 1

OUTPUT #1

```
[i - 1 for i in l]
```

High lexical overlap with the solution

[0, 1, 2] ☒

TEST

```
l = [1, 2, 3]
```

OUTPUT #2

```
ret_list = []
for e in l:
    ret_list.append(e + 1)
return ret_list
```

Low lexical overlap with the canonical solution,
But actually correct

[2, 3, 4] ✓

SOLUTION

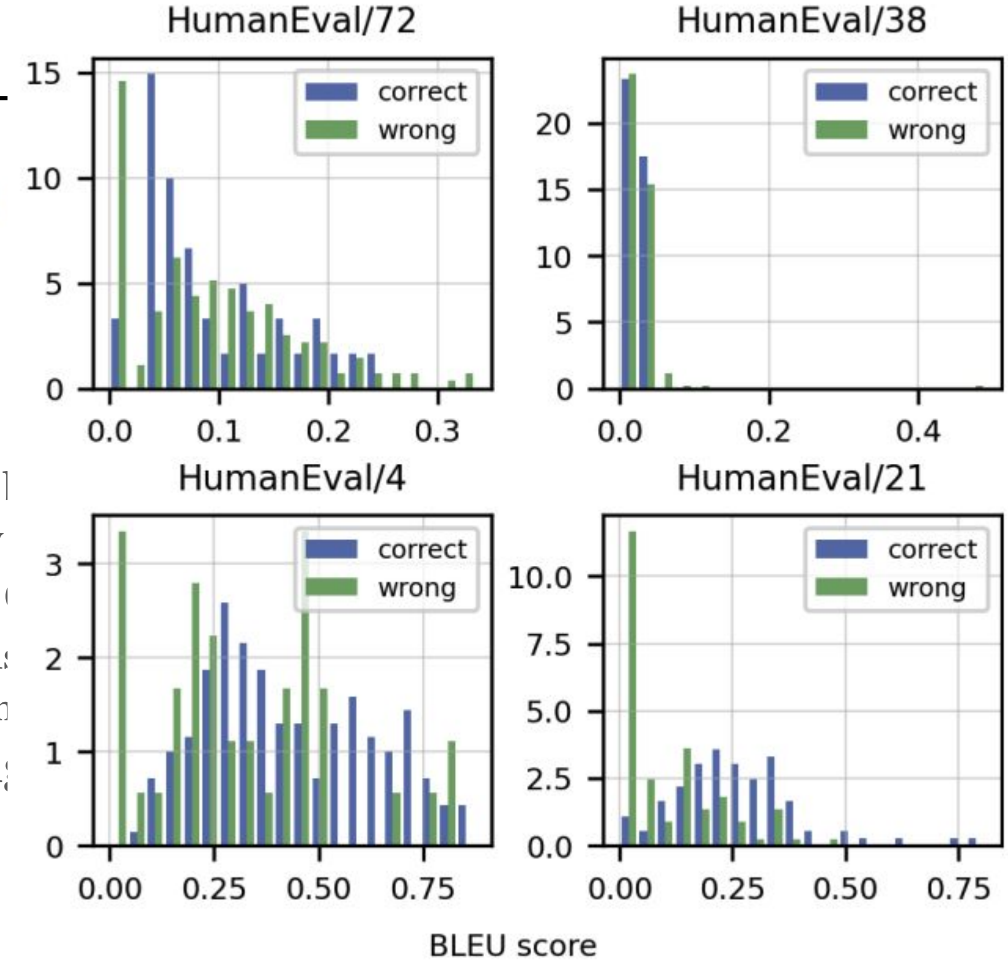
```
[i + 1 for i in l]
```

HumanEval Benchmark -

Evaluation: test case execution

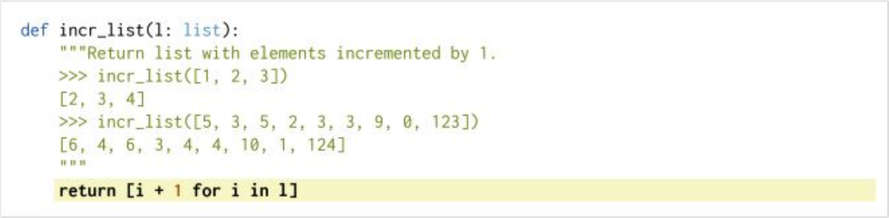
164 hand-written examples

- Why human-written?
  - “It is important for these tasks to l
    fraction of GitHub, which already
- Safety: Sandbox for Executing (
  - “Since publicly available programs
    incorrect, executing these progran
- Optimizing BLEU != Improving

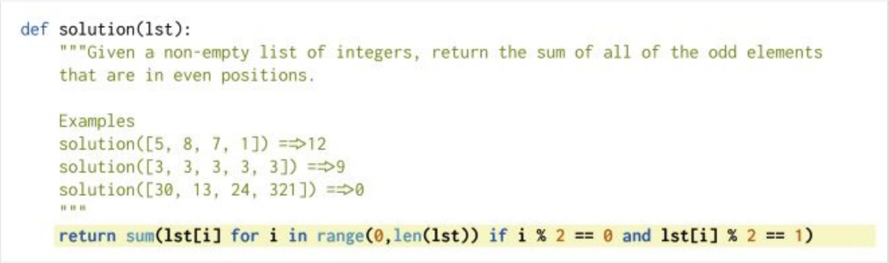


HumanEval Looks Like Toy Examples?

HumanEval Examples

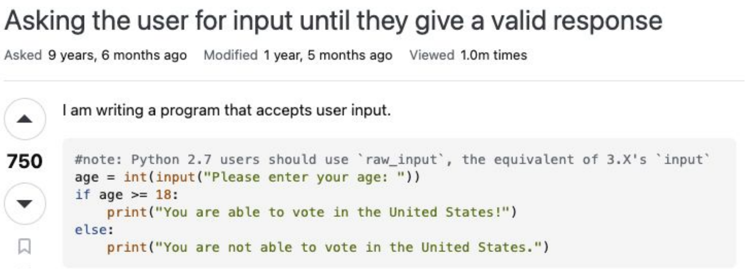


```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```



```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

Real-World Development Code



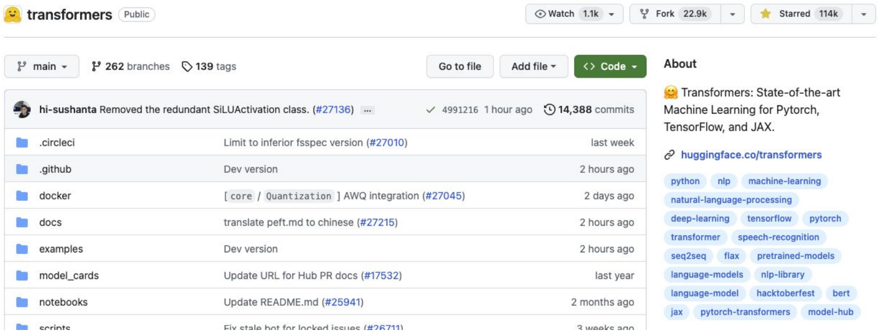
Asking the user for input until they give a valid response

Asked 9 years, 6 months ago Modified 1 year, 5 months ago Viewed 1.0m times

750

I am writing a program that accepts user input.

```python
#note: Python 2.7 users should use `raw_input`, the equivalent of 3.X's `input`
age = int(input("Please enter your age: "))
if age >= 18:
    print("You are able to vote in the United States!")
else:
    print("You are not able to vote in the United States.")
```



hi-sushanta Removed the redundant SiLUActivation class. (#27136) ... ✓ 4991216 1 hour ago 14,388 commits

.circleci	Limit to inferior fsspec version (#27010)	last week
.github	Dev version	2 hours ago
docker	[core / Quantization] AWQ integration (#27045)	2 days ago
docs	translate peft.md to chinese (#27215)	2 hours ago
examples	Dev version	2 hours ago
model_cards	Update URL for Hub PR docs (#17532)	last year
notebooks	Update README.md (#25941)	2 months ago
scripts	Fix stale bot for locked issues (#26711)	3 weeks ago

About

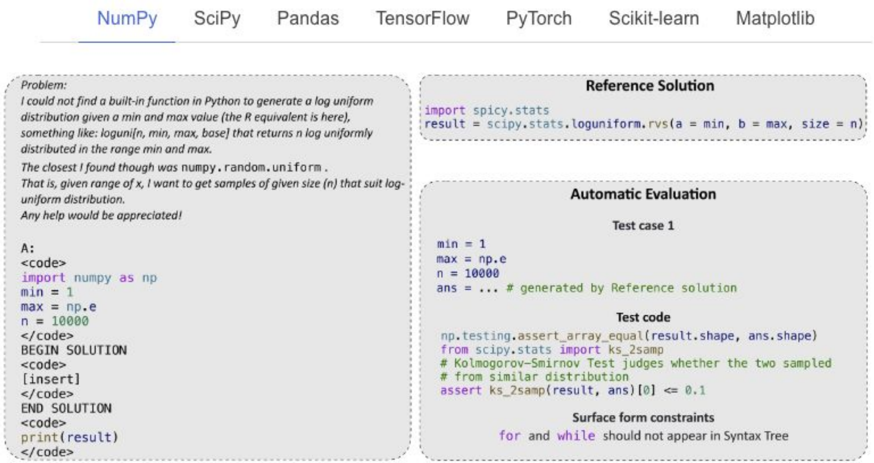
🤗 Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX.

huggingface.co/transformers

python nlp machine-learning natural-language-processing deep-learning tensorflow pytorch transformer speech-recognition seq2seq flax pretrained-models language-models nlp-library language-model hacktoberfest bert jax pytorch-transformers model-hub

Domain Exploration

- Leetcode Style: HumanEval, APPS, MBPP
    - Manually written or collected from code contest websites
    - Only uses Python built-in grammar
- Limited Domains: e.g., Data Science
    - DS-1000: StackOverflow questions
    - ARCADE: Interactive Jupyter Notebooks
    - ExeDS
    - … …
- Open Domain: ODEX
    - 79 Python libraries
    - Four natural languages



Problem:

I could not find a built-in function in Python to generate a log uniform distribution given a min and max value (the R equivalent is here), something like: loguni[n, min, max, base] that returns n log uniformly distributed in the range min and max.

The closest I found though was `numpy.random.uniform`.

That is, given range of x, I want to get samples of given size (n) that suit log-uniform distribution.

Any help would be appreciated!

```
A:
<code>
import numpy as np
min = 1
max = np.e
n = 10000
</code>
BEGIN SOLUTION
<code>
[insert]
</code>
END SOLUTION
<code>
print(result)
</code>
```

Reference Solution

```
import spicy.stats
result = scipy.stats.loguniform.rvs(a = min, b = max, size = n)
```

Automatic Evaluation

Test case 1

```
min = 1
max = np.e
n = 10000
ans = ... # generated by Reference solution
```

Test code

```
np.testing.assert_array_equal(result.shape, ans.shape)
from scipy.stats import ks_2samp
# Kolmogorov-Smirnov Test judges whether the two sampled
# from similar distribution
assert ks_2samp(result, ans)[0] <= 0.1
```

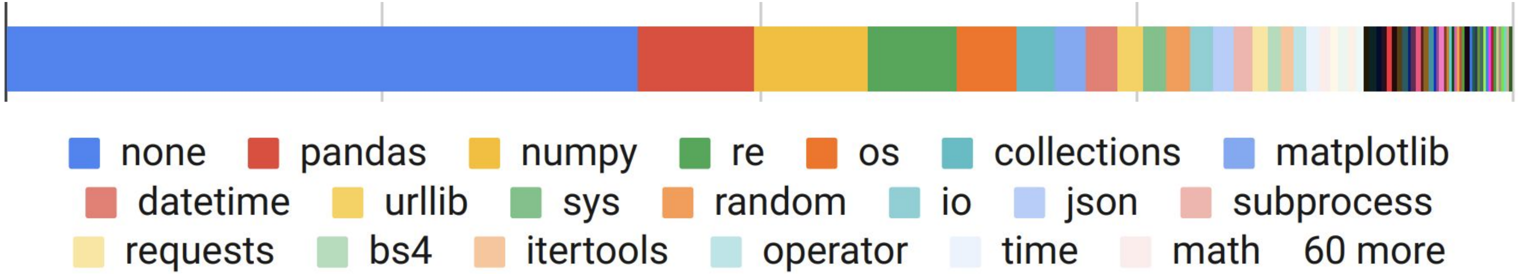
Surface form constraints

`for` and `while` should not appear in Syntax Tree

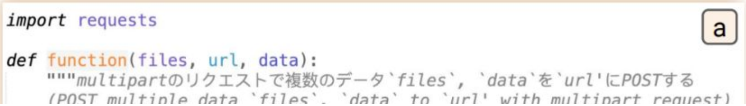
NumPy example problem involving randomness, requiring the use of a specialist knowledge test.

Execution-Based Evaluation for Open-Domain Code Generation

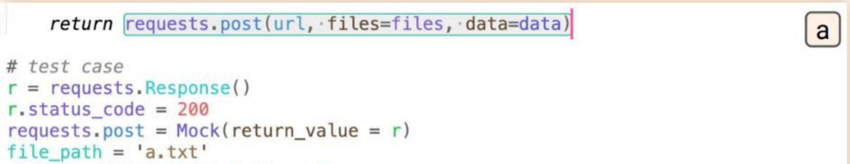
- Larger Domain Coverage



- Test execution on real-world coding queries
  - Collected from StackOverflow questions
- Support four natural languages as input
  - English, Spanish, Japanese, Russian

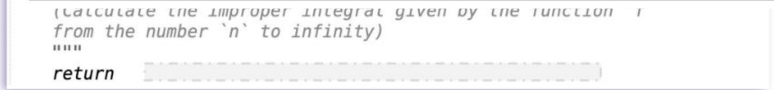


```
import requests

def function(files, url, data):
    """multipartのリクエストで複数のデータ`files`, `data`を`url'にPOSTする
    (POST multiple data `files`, `data` to `url' with multipart request)
```

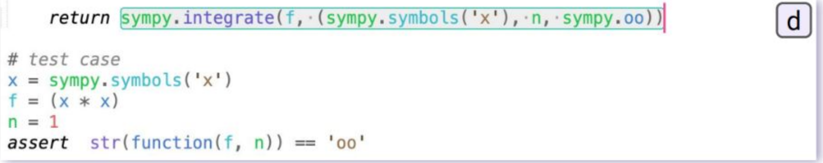


```
    return requests.post(url, files=files, data=data)

# test case
r = requests.Response()
r.status_code = 200
requests.post = Mock(return_value = r)
file_path = 'a.txt'
```

Dataset	Samples	Domain	Executable?	Avg. Test Cases	Data Source	NL
JuICe (Agashe et al., 2019)	1,981	open	✗	-	GitHub Notebooks	en
HumanEval (Chen et al., 2021)	164	4	✓	7.7	Hand-written	en
MBPP (Austin et al., 2021)	974	8	✓	3.0	Hand-written	en
APPS (Hendrycks et al., 2021)	10,000	0	✓	13.2	Competitions	en
DSP (Chandel et al., 2022)	1,119	16	✓	2.1	Github Notebooks	en
MTPB (Nijkamp et al., 2022)	115	8	✓	5.0	Hand-written	en
Exe-DS (Huang et al., 2022)	534	28	✓	-	GitHub Notebooks	en
DS-1000 (Lai et al., 2022)	1,000	7	✓	1.6	StackOverflow	en
CoNaLa (Yin et al., 2018)	2,879	open	✗	-	StackOverflow	en
MCoNaLa (Wang et al., 2022)	896	open	✗	-	StackOverflow	es, ja, ru
ODEX	945	79	✓	1.8	StackOverflow Hand-Written	en, es, ja, ru



```
    (calculate the improper integral given by the function `f`
    from the number `n` to infinity)
    """
    return
```



```
    return sympy.integrate(f, (sympy.symbols('x'), n, sympy.oo))

# test case
x = sympy.symbols('x')
f = (x * x)
n = 1
assert  str(function(f, n)) == 'oo'
```

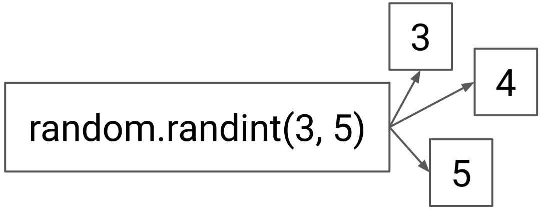
ODEX: Unique Challenges for Execution

Closed-domain code: easy to execute and verify

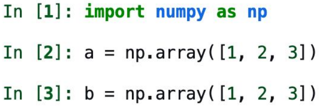
```
assert func([1, 2, 10]) == [2, 3, 11]
```

Open-domain code:

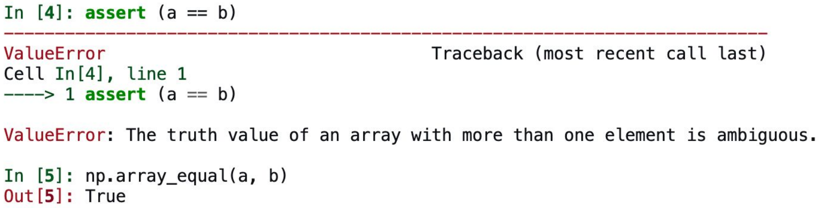
- Random outputs



- Specialized verification



```
In [1]: import numpy as np

In [2]: a = np.array([1, 2, 3])

In [3]: b = np.array([1, 2, 3])
```



```
In [4]: assert (a == b)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[4], line 1
----> 1 assert (a == b)

ValueError: The truth value of an array with more than one element is ambiguous.

In [5]: np.array_equal(a, b)
Out[5]: True
```

- (Potentially) not reproducible queries
  - HTTP requests, e.g., requests.post("https://def.xyz", data={'key': 'value'})

Significant Performance Gaps: Open vs. Closed

- Although Codex performs better overall
- CodeGen has smaller domain gaps

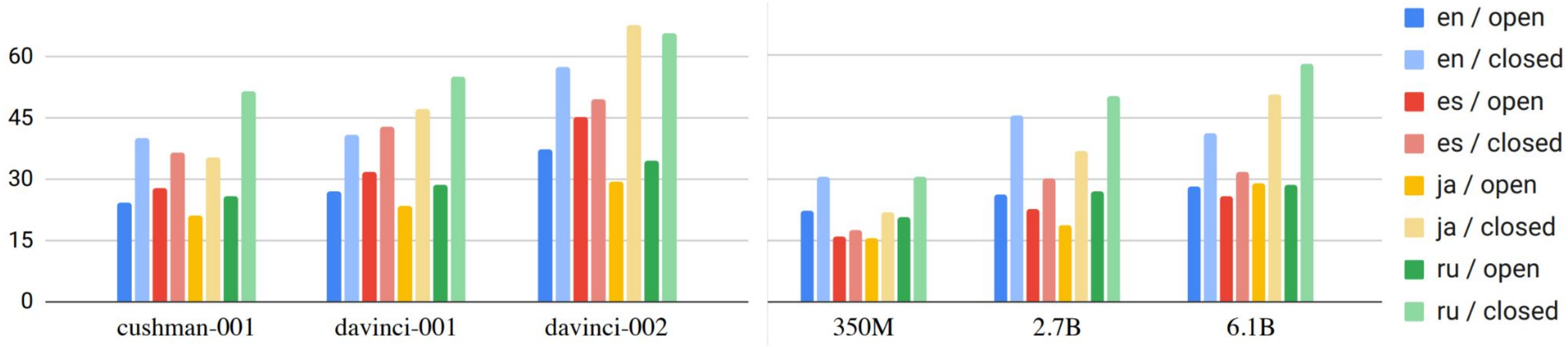
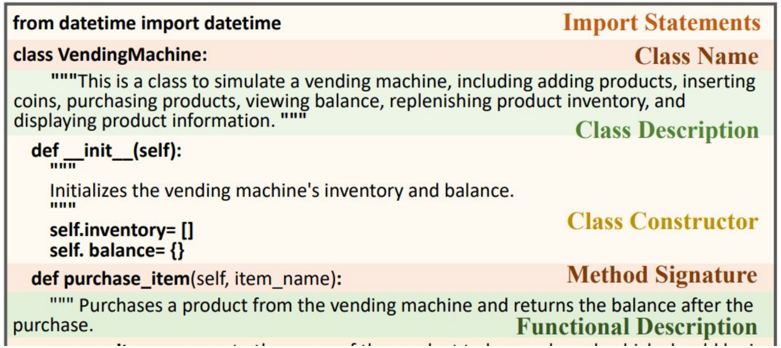


Figure 7: CODEX (left) and CODEGEN (right) *pass@1* on open- and closed-domain problems in each language.

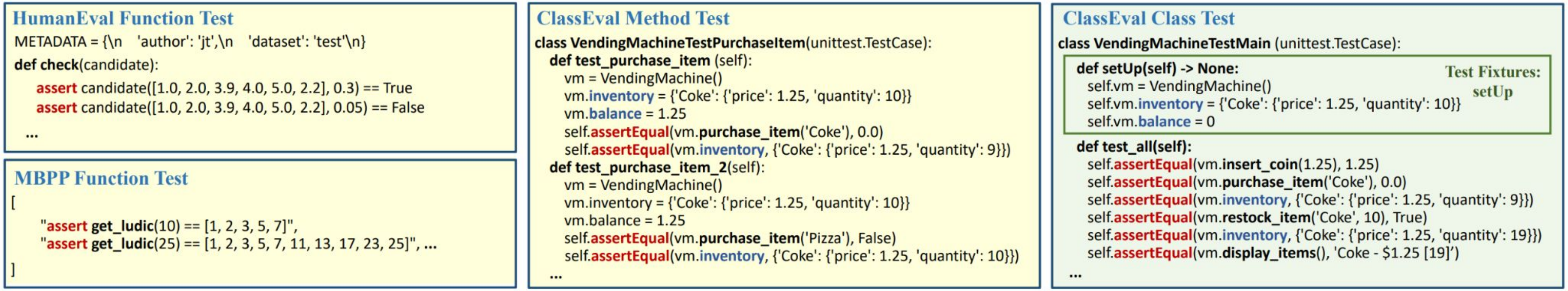
Functional Complexity

- Function Level: HumanEval, MBPP
- Class Level: ClassEval



```
from datetime import datetime
class VendingMachine:
    """This is a class to simulate a vending machine, including adding products, inserting
coins, purchasing products, viewing balance, replenishing product inventory, and
displaying product information. """
  def __init__(self):
    """
    Initializes the vending machine's inventory and balance.
    """
    self.inventory= []
    self. balance= {}
  def purchase_item(self, item_name):
    """ Purchases a product from the vending machine and returns the balance after the
purchase.
```

Import Statements
Class Name
Class Description
Class Constructor
Method Signature
Functional Description



HumanEval Function Test

```
METADATA = {\n    'author': 'jt',\n    'dataset': 'test'\n}
def check(candidate):
    assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.3) == True
    assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.05) == False
  ...
```

MBPP Function Test

```
[
    "assert get_ludic(10) == [1, 2, 3, 5, 7]",
    "assert get_ludic(25) == [1, 2, 3, 5, 7, 11, 13, 17, 23, 25]", ...
]
```

ClassEval Method Test

```
class VendingMachineTestPurchaseItem(unittest.TestCase):
  def test_purchase_item (self):
    vm = VendingMachine()
    vm.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}
    vm.balance = 1.25
    self.assertEqual(vm.purchase_item('Coke'), 0.0)
    self.assertEqual(vm.inventory, {'Coke': {'price': 1.25, 'quantity': 9}})
  def test_purchase_item_2(self):
    vm = VendingMachine()
    vm.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}
    vm.balance = 1.25
    self.assertEqual(vm.purchase_item('Pizza'), False)
    self.assertEqual(vm.inventory, {'Coke': {'price': 1.25, 'quantity': 10}})
  ...
```

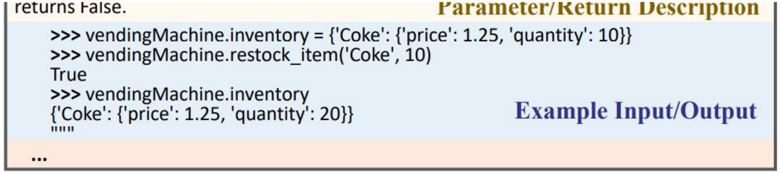
ClassEval Class Test

```
class VendingMachineTestMain (unittest.TestCase):
  def setUp(self) -> None:
    self.vm = VendingMachine()
    self.vm.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}
    self.vm.balance = 0
  def test_all(self):
    self.assertEqual(vm.insert_coin(1.25), 1.25)
    self.assertEqual(vm.purchase_item('Coke'), 0.0)
    self.assertEqual(vm.inventory, {'Coke': {'price': 1.25, 'quantity': 9}})
    self.assertEqual(vm.restock_item('Coke', 10), True)
    self.assertEqual(vm.inventory, {'Coke': {'price': 1.25, 'quantity': 19}})
    self.assertEqual(vm.display_items(), 'Coke - $1.25 [19]')
  ...
```

Figure 4: Test Cases in Existing Benchmarks and ClassEval

"Write a python function to find the first repeated character in a given string."

Figure 1: Examples in Existing Benchmarks



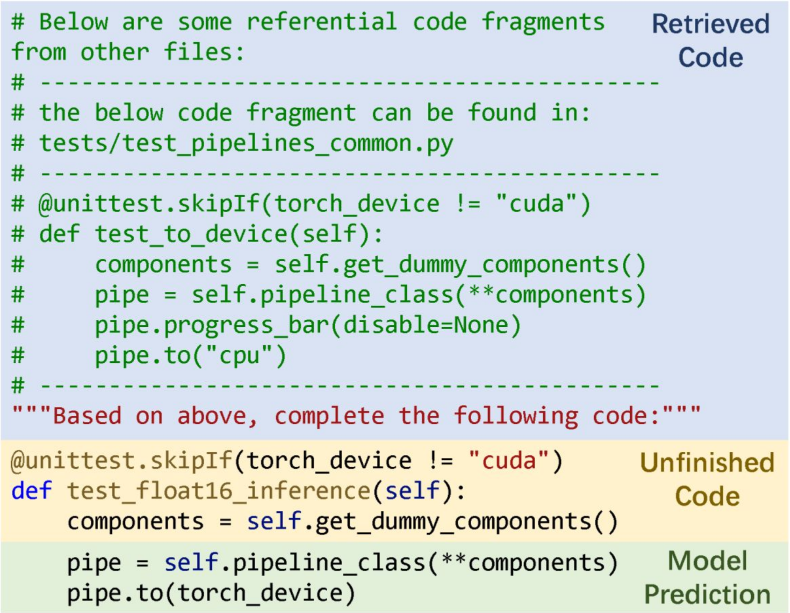
```
returns False.
    >>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}
    >>> vendingMachine.restock_item('Coke', 10)
    True
    >>> vendingMachine.inventory
    {'Coke': {'price': 1.25, 'quantity': 20}}
    """
  ...
```

Parameter/Return Description
Example Input/Output

Figure 2: An Example of Class Skeleton in ClassEval

Functional Complexity

- Function Level: HumanEval, MBPP
- Class Level: ClassEval
- Repository Level:
    - RepoCoder
        - Retrieval-augmented generation
        - Multiple iterations
    - RepoEval
        - Collected 14 Github Repositories
        - Metrics:
            - exact match
            - exact similarity
            - execution

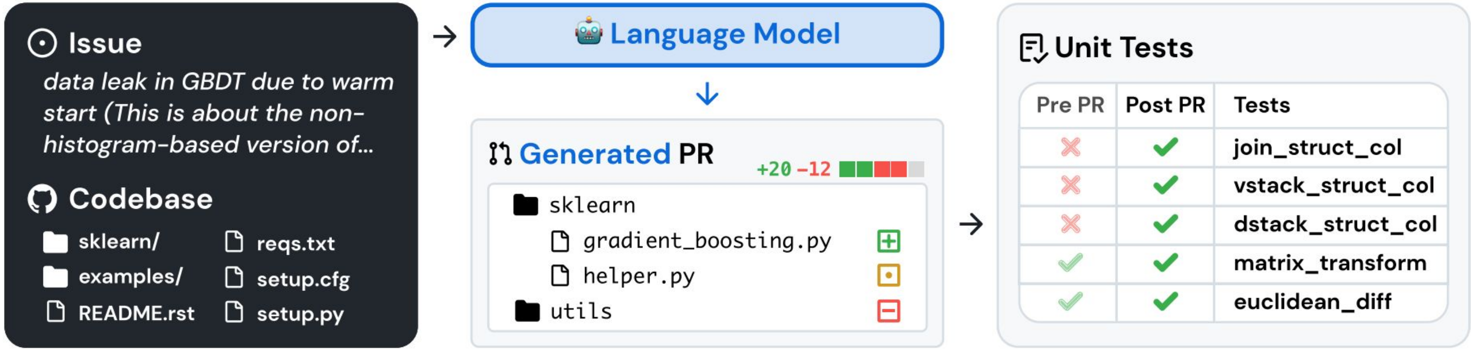


```
# Below are some referential code fragments
from other files:
# ---------------------------------------------
# the below code fragment can be found in:
# tests/test_pipelines_common.py
# ---------------------------------------------
# @unittest.skipIf(torch_device != "cuda")
# def test_to_device(self):
#     components = self.get_dummy_components()
#     pipe = self.pipeline_class(**components)
#     pipe.progress_bar(disable=None)
#     pipe.to("cpu")
# ---------------------------------------------
"""Based on above, complete the following code:"""
@unittest.skipIf(torch_device != "cuda")
def test_float16_inference(self):
    components = self.get_dummy_components()
    pipe = self.pipeline_class(**components)
    pipe.to(torch_device)
```

Figure 3: A visual example demonstrating the format of the RepoCoder prompt, which combines the retrieved code snippets from the repository with the unfinished code present in the target file.

Functional Complexity

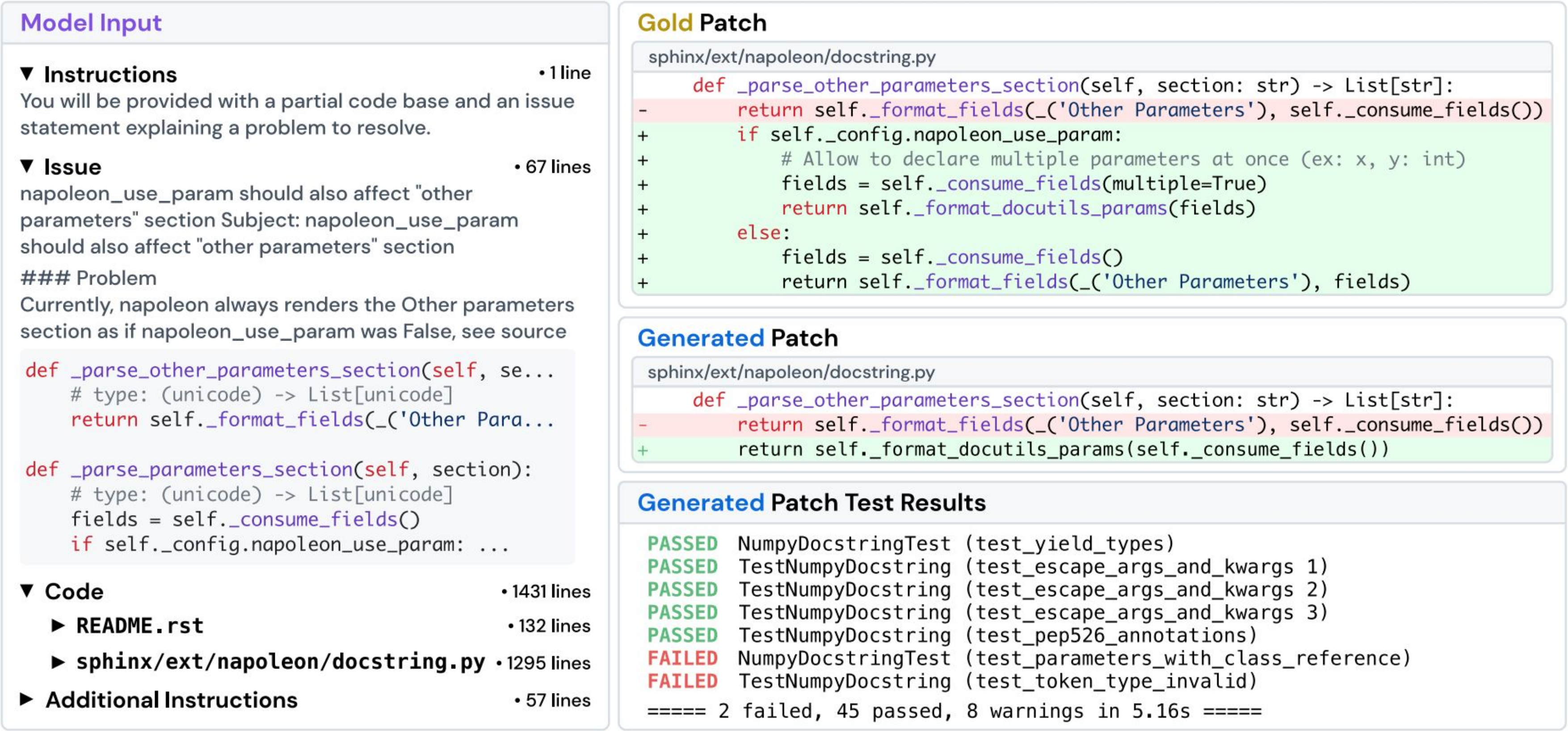
- Function Level: HumanEval, MBPP
- Class Level: ClassEval
- Repository Level: RepoCoder & RepoEval
- Repo-Level Pull Requests: SWE-Bench



data leak in GBDT due to warm start (This is about the non-histogram-based version of...

Pre PR	Post PR	Tests
✗	✓	join_struct_col
✗	✓	vstack_struct_col
✗	✓	dstack_struct_col
✓	✓	matrix_transform
✓	✓	euclidean_diff

Figure 1: SWE-bench sources task instances from real-world Python repositories by connecting GitHub issues to merged pull request solutions that resolve related tests. Provided with the issue text and a codebase snapshot, models generate a patch that is evaluated against real tests.



Model Input

▼ Instructions • 1 line

You will be provided with a partial code base and an issue statement explaining a problem to resolve.

▼ Issue • 67 lines

napoleon_use_param should also affect "other parameters" section Subject: napoleon_use_param should also affect "other parameters" section

Problem

Currently, napoleon always renders the Other parameters section as if napoleon_use_param was False, see source

```
def _parse_other_parameters_section(self, se...
    # type: (unicode) -> List[unicode]
    return self._format_fields(_('Other Para...

def _parse_parameters_section(self, section):
    # type: (unicode) -> List[unicode]
    fields = self._consume_fields()
    if self._config.napoleon_use_param: ...
```

▼ Code • 1431 lines

▶ README.rst • 132 lines

▶ sphinx/ext/napoleon/docstring.py • 1295 lines

▶ Additional Instructions • 57 lines

Gold Patch

sphinx/ext/napoleon/docstring.py

```
     def _parse_other_parameters_section(self, section: str) -> List[str]:
-        return self._format_fields(_('Other Parameters'), self._consume_fields())
+        if self._config.napoleon_use_param:
+            # Allow to declare multiple parameters at once (ex: x, y: int)
+            fields = self._consume_fields(multiple=True)
+            return self._format_docutils_params(fields)
+        else:
+            fields = self._consume_fields()
+            return self._format_fields(_('Other Parameters'), fields)
```

Generated Patch

sphinx/ext/napoleon/docstring.py

```
     def _parse_other_parameters_section(self, section: str) -> List[str]:
-        return self._format_fields(_('Other Parameters'), self._consume_fields())
+        return self._format_docutils_params(self._consume_fields())
```

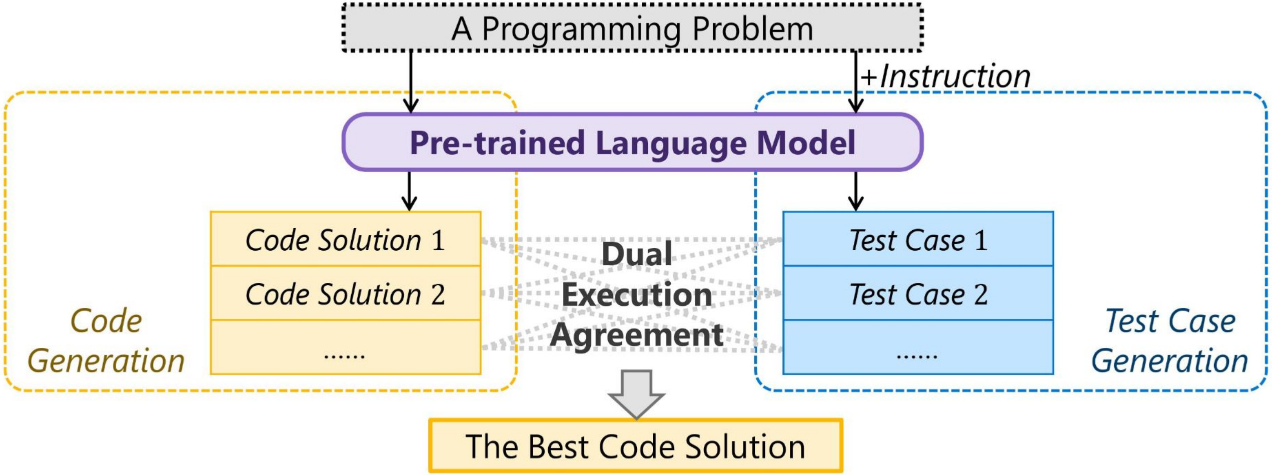
Generated Patch Test Results

```
PASSED  NumpyDocstringTest (test_yield_types)
PASSED  TestNumpyDocstring (test_escape_args_and_kwargs 1)
PASSED  TestNumpyDocstring (test_escape_args_and_kwargs 2)
PASSED  TestNumpyDocstring (test_escape_args_and_kwargs 3)
PASSED  TestNumpyDocstring (test_pep526_annotations)
FAILED  NumpyDocstringTest (test_parameters_with_class_reference)
FAILED  TestNumpyDocstring (test_token_type_invalid)
===== 2 failed, 45 passed, 8 warnings in 5.16s =====
```

Figure 6: We show an example of an formatted task instance, a model prediction, and the testing framework logs. Results and inputs are stylized for readability. In the gold and generated patch file, red-highlighted lines represent deletions and green-highlighted lines represent additions.

Automated & Improved Testing

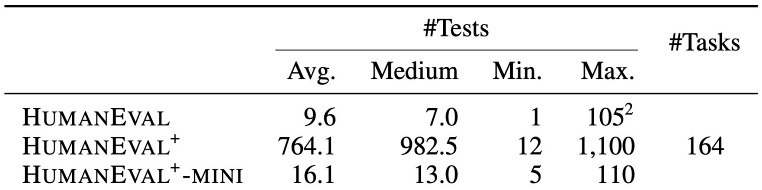
- Initial approach: human (expert) write test cases
- Evaluate on code without annotated tests? CodeT
  - Models generate solutions and test cases at the same time
  - Cross-validation between multiple solutions and tests



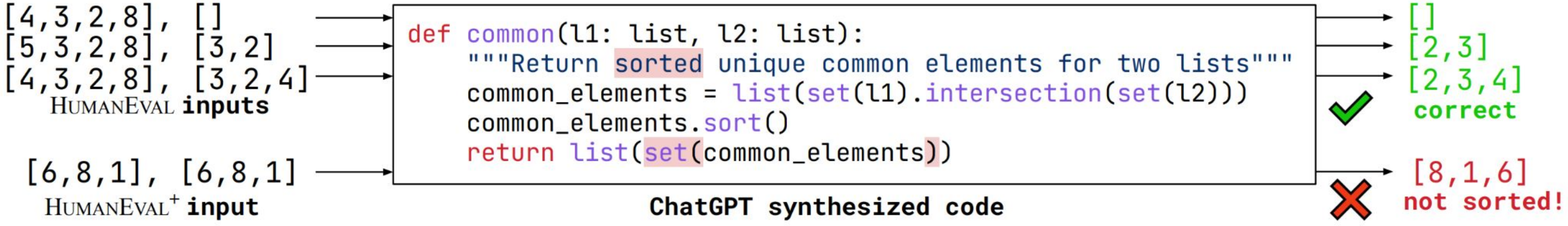
Automated & Improved Testing

- Initial approach: human (expert) write test cases
- LLMs to create test cases → EvalPlus
    - ○ Scale-up test generation, reduce human effort
    - ○ More sufficient tests, can find wrong solutions

Table 2: Overview of EvalPlus-improved benchmarks.



	#Tests				#Tasks
	Avg.	Medium	Min.	Max.	
HumanEval	9.6	7.0	1	105[2]	164
HumanEval$^+$	764.1	982.5	12	1,100	
HumanEval$^+$-mini	16.1	13.0	5	110	



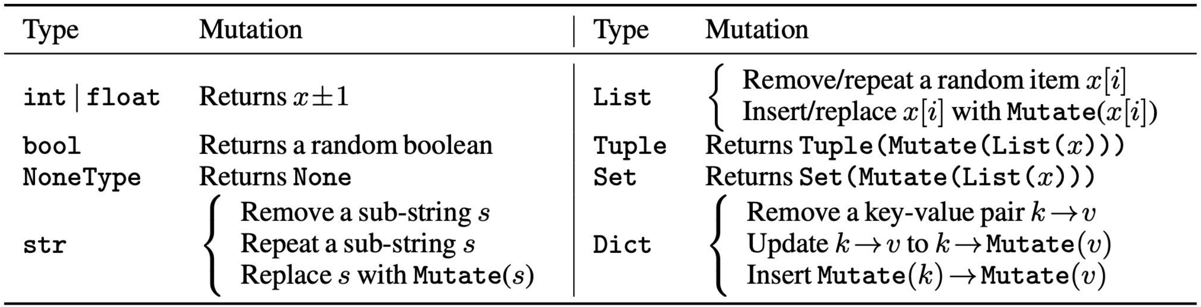
```python
def common(l1: list, l2: list):
    """Return sorted unique common elements for two lists"""
    common_elements = list(set(l1).intersection(set(l2)))
    common_elements.sort()
    return list(set(common_elements))
```

Figure 1: Exemplary wrong code synthesized by ChatGPT for HumanEval #58.

Automated & Improved Testing

- Initial approach: human (expert) write test cases
- LLMs to create test cases → EvalPlus
- Fuzzing: a common method in software engineering
  - Type-aware value mutations / alterations
  - Limitation: can only apply to Python basic types, not open-domain ones, e.g., numpy array

Table 1: List of basic type-aware mutations over input x.



Type	Mutation	Type	Mutation
int | float	Returns $x \pm 1$	List	Remove/repeat a random item $x[i]$ Insert/replace $x[i]$ with Mutate($x[i]$)
bool	Returns a random boolean	Tuple	Returns Tuple(Mutate(List(x)))
NoneType	Returns None	Set	Returns Set(Mutate(List(x)))
str	Remove a sub-string s Repeat a sub-string s Replace s with Mutate(s)	Dict	Remove a key-value pair $k \to v$ Update $k \to v$ to $k \to$ Mutate(v) Insert Mutate(k) $\to$ Mutate(v)