Evaluation: Metrics and Benchmarks

Daniel Fried

11-891: Neural Code Generation

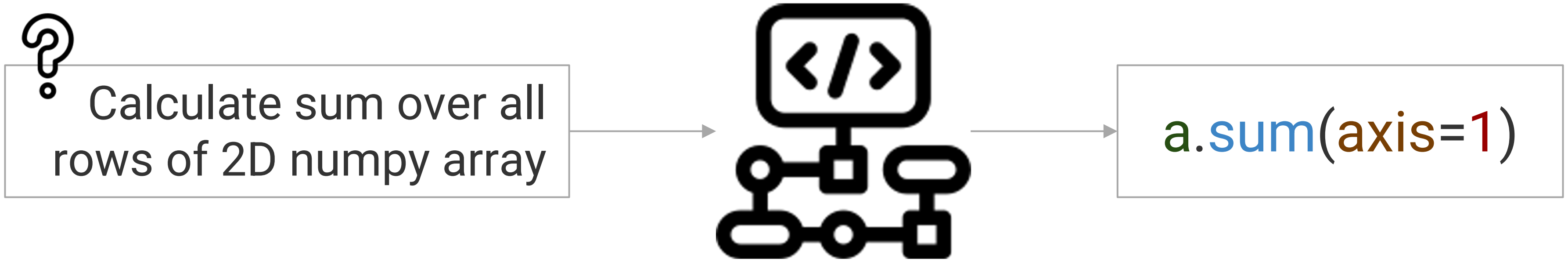
https://cmu-codegen.github.io/s2024/

Language
Technologies
Institute

With slides from Zora Wang and Nikitha Rao

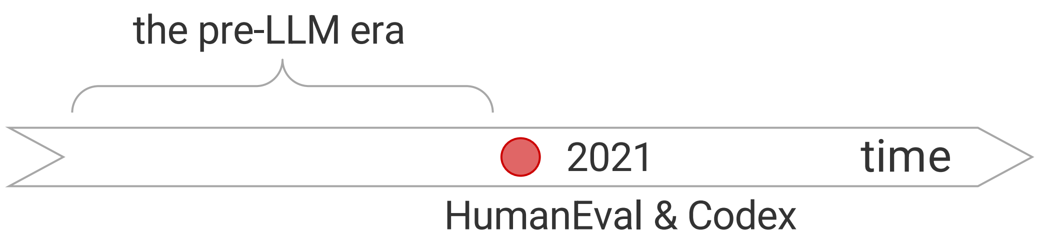
The NL2Code Task

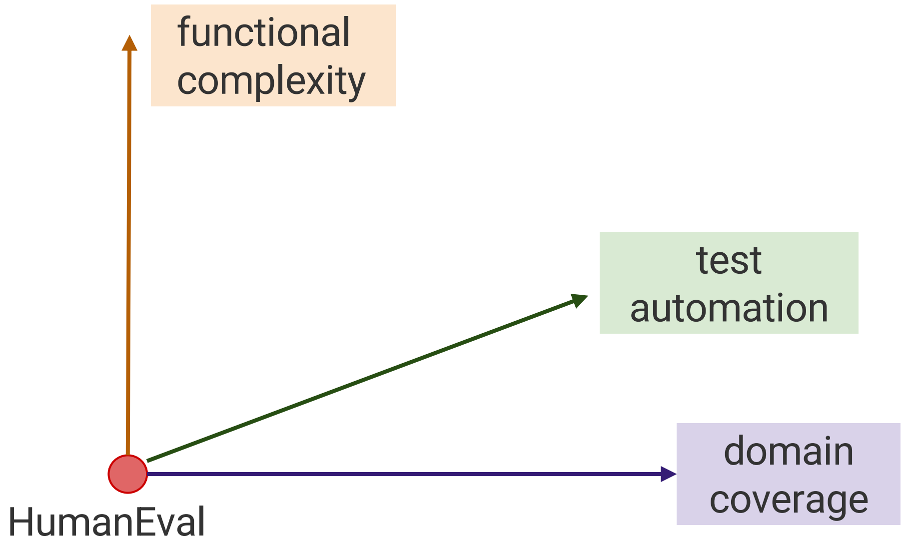
- Given a natural language instruction Q, generate code implementation C



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





Pre-2020

- Most code snippets were short, and evaluated using BLEU or exact match.
- Datasets were fairly large, with dedicated training sets.

Natural Language	Bash Command(s)

find .java files in the current directory tree that contain the pattern 'TODO' and print their names	`grep -l "TODO" *.java`
	`find . -name "*.java" -exec grep -il "TODO" {} \;`
	`find . -name "*.java" | xargs -I {} grep -l "TODO" {}`

display the 5 largest files in the current directory and its sub-directories	`find .  -type f | sort -nk 5,5 | tail -5`
	`du -a . | sort -rh | head -n5`
	`find . -type f -printf '%s %p\n' | sort -rn | head -n5`

search for all jpg images on the system and archive them to tar ball "images.tar"	`tar -cvf images.tar $(find / -type f -name *.jpg)`
	`tar -rvf images.tar $(find / -type f -name *.jpg)`
	`find / -type f -name "*.jpg" -exec tar -cvf images.tar {} \;`

	Train	Dev	Test
# pairs	8,090	609	606
# unique nls	7,340	549	547

NL2Bash. Lin et al. 2018.

Pre-2020

- Most code snippets were short, and evaluated using BLEU or exact match.
- Datasets were fairly large, with dedicated training sets.

Dataset	PL	# pairs	# words	# tokens	Avg. # w. in nl	Avg. # t. in code	NL collection	Code collection	Semantic alignment	Introduced by
IFTTT	DSL	86,960	–	–	7.0	21.8	scraped	scraped	Noisy	(Quirk et al., 2015)
C#2NL*	C#	66,015	24,857	91,156	12	38				(Iyer et al., 2016)
SQL2NL*	SQL	32,337	10,086	1,287	9	46				
RegexLib	Regex	3,619	13,491	179✱	36.4	58.8✱			Good✱	(Zhong et al., 2018)
HeartStone	Python	665	–	–	7	352✱	game card description	game card source code		(Ling et al., 2016)
MTG	Java	13,297	–	–	21	1,080✱				
StaQC	Python	147,546	17,635	137,123	9	86	extracted using ML	extracted using ML	Noisy	(Yao et al., 2018)
	SQL	119,519	9,920	21,413	9	60				
NL2RX	Regex	10,000	560	45✱†	10.6	26✱	synthesized & paraphrased	synthesized	Very Good	(Locascio et al., 2016)
WikiSQL	SQL	80,654	–	–	–	–				(Zhong et al., 2017)
NLMAPS	DSL	2,380	1,014	–	10.9	16.0	synthesized given code	expert written	Very Good	(Haas and Riezler, 2016)
Jobs640★	DSL	640	391	58†	9.8	22.9	user written	expert written given NL		(Tang and Mooney, 2001)
GEO880	DSL	880	284	60†	7.6	19.1				(Zelle and Mooney, 1996)
Freebase917	DSL	917	–	–	–	–				(Cai and Yates, 2013)
ATIS★	DSL	5,410	936	176†	11.1	28.1				(Dahl et al., 1994)
WebQSP	DSL	4,737	–	–	–	–	search log			(Yih et al., 2016)
NL2RX-KB13	Regex	824	715	85✱†	7.1	19.0✱	turker written			(Kushman and Barzilay, 2013)
Django☆	Python	18,805	–	–	14.3	–	expert written given code	scraped		(Oda et al., 2015)
NL2Bash	Bash	9,305	7,790	6,234	11.7	7.7				Ours

NL2Bash. Lin et al. 2018.

Evaluation Metrics

Reference Matching: BLEU

- Developed for machine translation (Papineni et al. 2002)
- Compares n-gram overlap between predicted and reference
- Typically, uses n-grams up to 4 (BLEU-4)

Reference: Taro visited Hanako

System: the Taro visited the Hanako

1-gram: 3/5
2-gram: 1/4

Brevity: min(1, |System|/|Reference|) = min(1, 5/3)

brevity penalty = 1.0

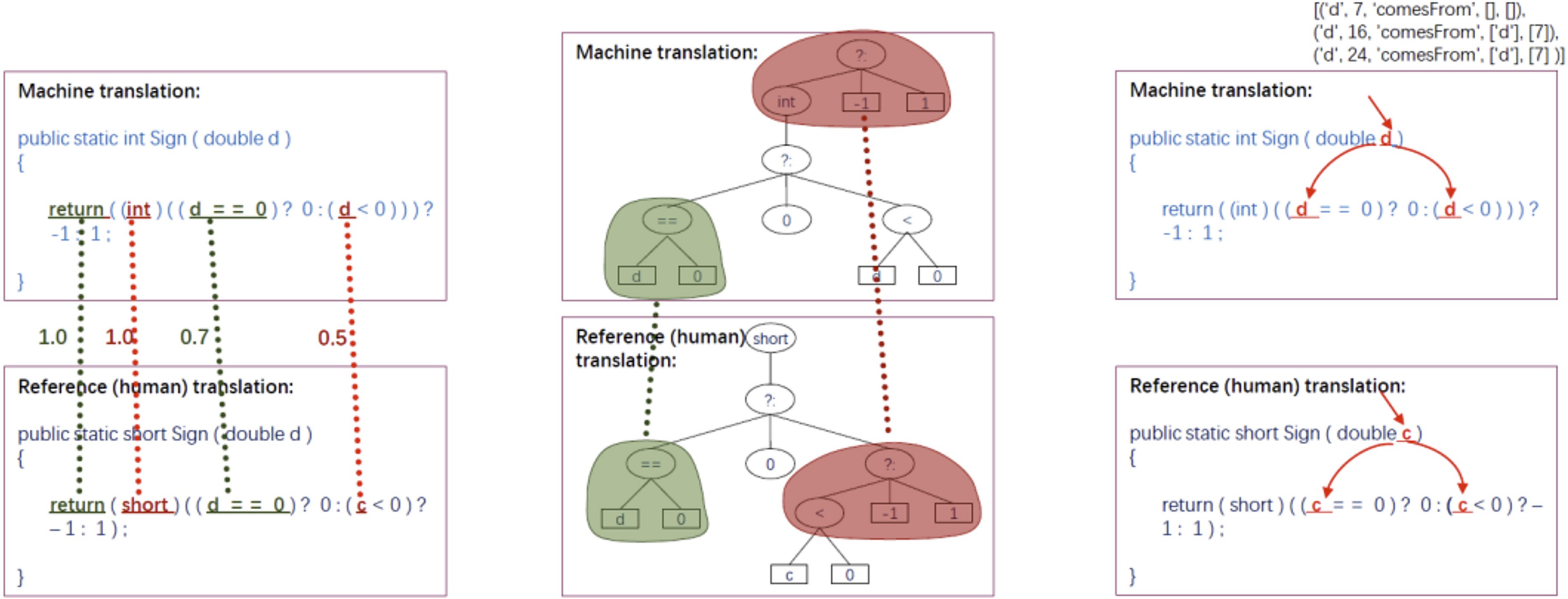
BLEU-2 = $(3/5*1/4)^{1/2}$ * 1.0
= 0.387

Example from Graham Neubig

Ren et al. 2020

Reference Matching: CodeBLEU

Higher weight for keywords | Match syntactic subtrees | Match data dependency graphs



Weighted N-Gram Match | Syntactic AST Match | Semantic Data-flow Match

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

Ren et al. 2020

Reference Matching: CodeBLEU

- When evaluating evaluation metrics, check correlation with human judgements.
- In CodeBLEU: rate code outputs on a Likert scale of general quality (1=very bad; 5=very good)

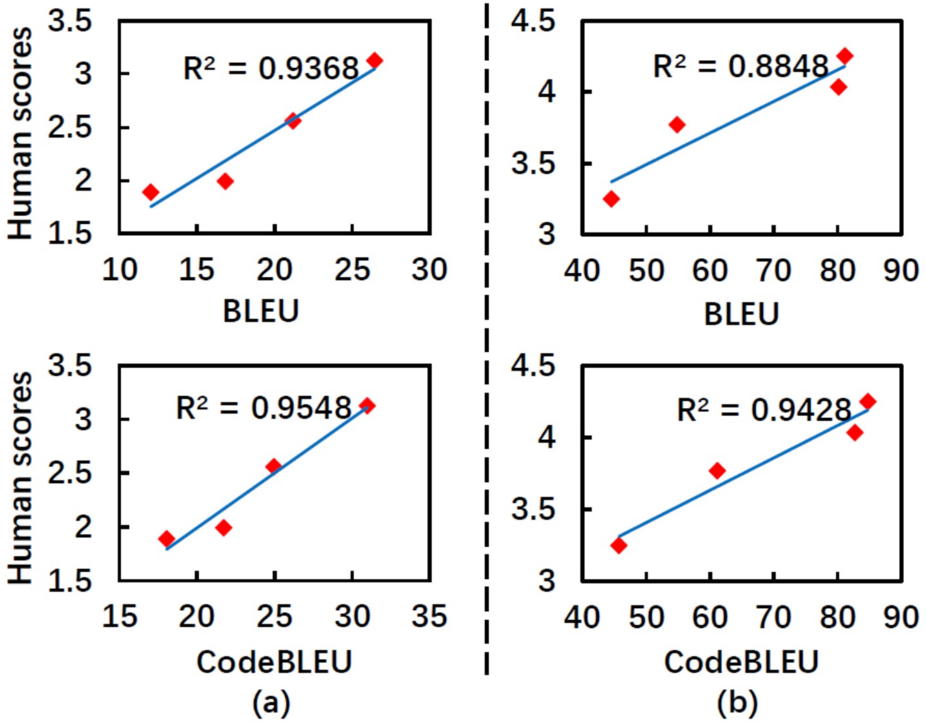
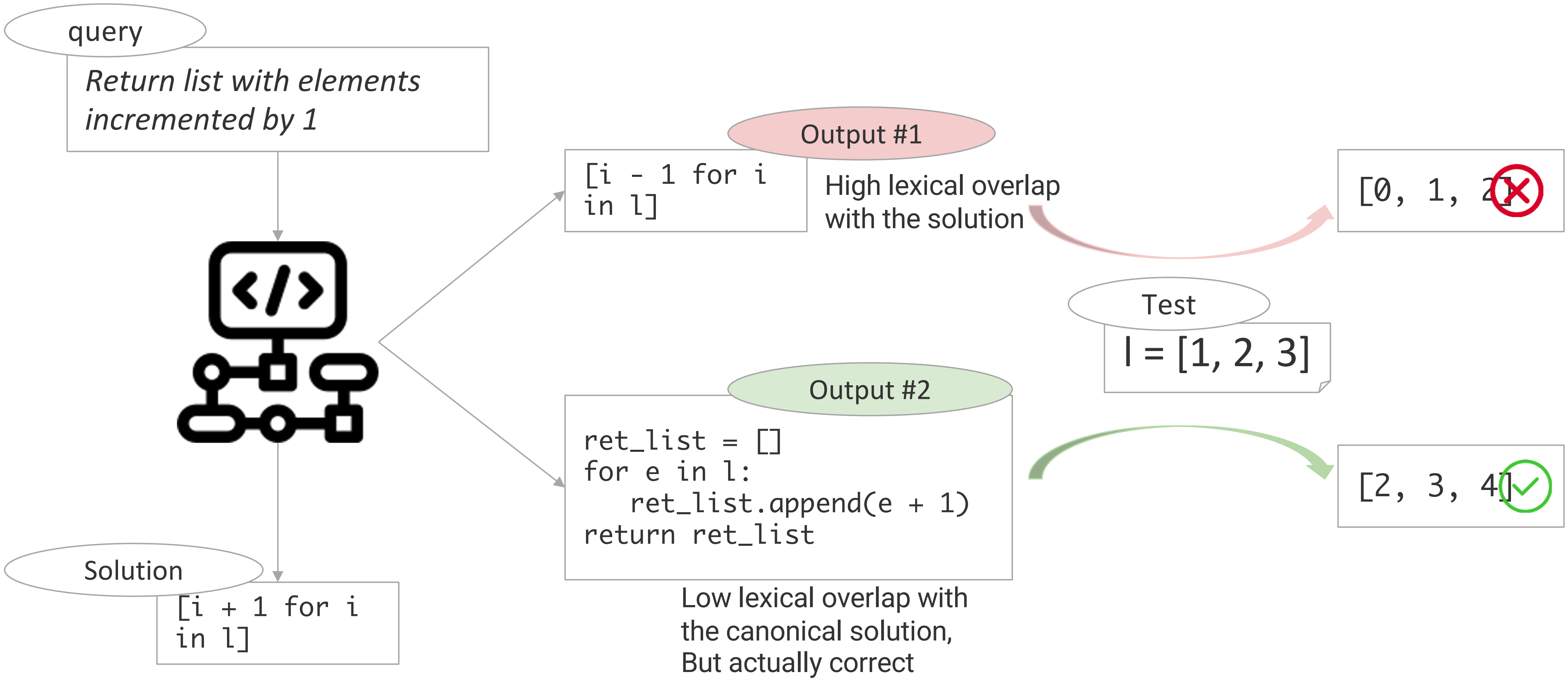


Figure 5: BLEU and CodeBLEU predict human evaluation scores. (a) Text-to-code; (b) Code translation.

Issues: Evaluations Are Not Rigorous



HumanEval Benchmark

- Evaluation: test case execution
- 164 hand-written examples, by authors of the paper
- Why human-written?
  - "It is important for these tasks to be hand-written, since our models are trained on a large fraction of GitHub, which already contains solutions to problems from a variety of sources. "

```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)
```

MBPP: Mostly Basic Python Programs

- Similar to HumanEval, but a bit easier
- 974 short Python problems, written by crowdworkers
  - 58% mathematical, 43% list processing, 19% string processing

Austin et al. 2021

MBPP: Mostly Basic Python Programs

- Model performance is sensitive to sampling temperature and number of candidates (similar findings in HumanEval/Codex paper)

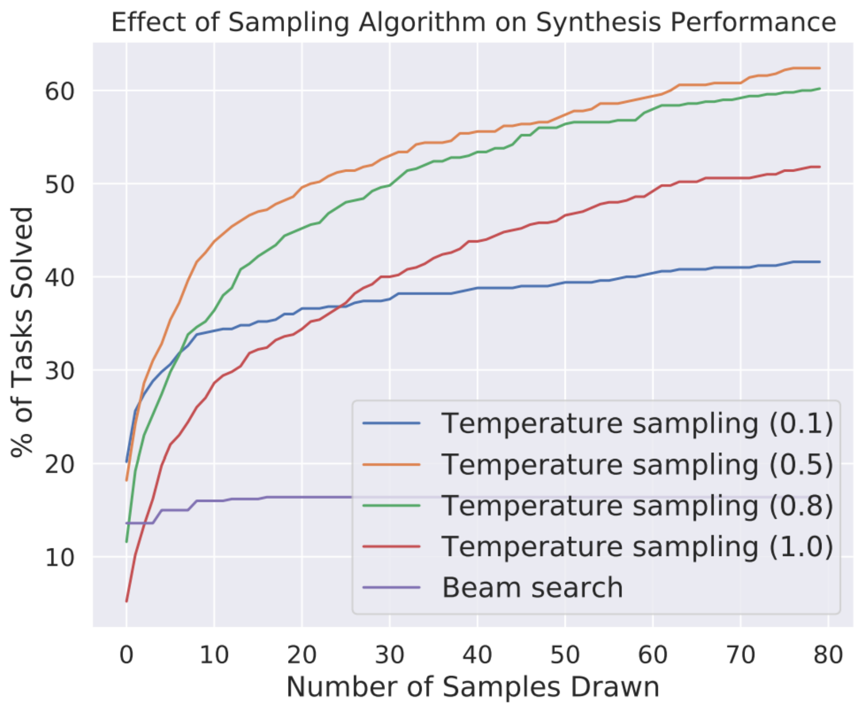


Figure 9: Higher temperatures achieve better scaling with more samples, but perform worse with a smaller budget.

Austin et al. 2021

MBPP: Mostly Basic Python Programs

- BLEU against a reference solution is uncorrelated with whether samples pass execution tests (similar findings in Codex paper).

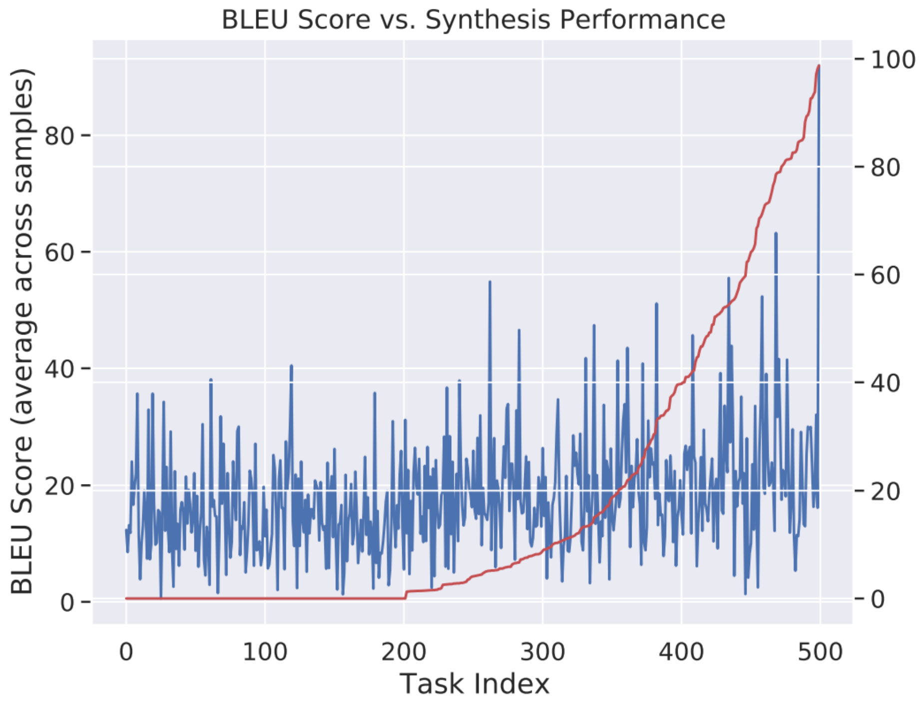


Figure 10: Comparison of BLEU score and synthesis performance for the 137B parameter model. No strong correlation is observed.

Austin et al. 2021

MBPP: Mostly Basic Python Programs

- Model evaluated is a large Google LLM, LaMDA, trained mostly on natural language, which has some interaction ability.

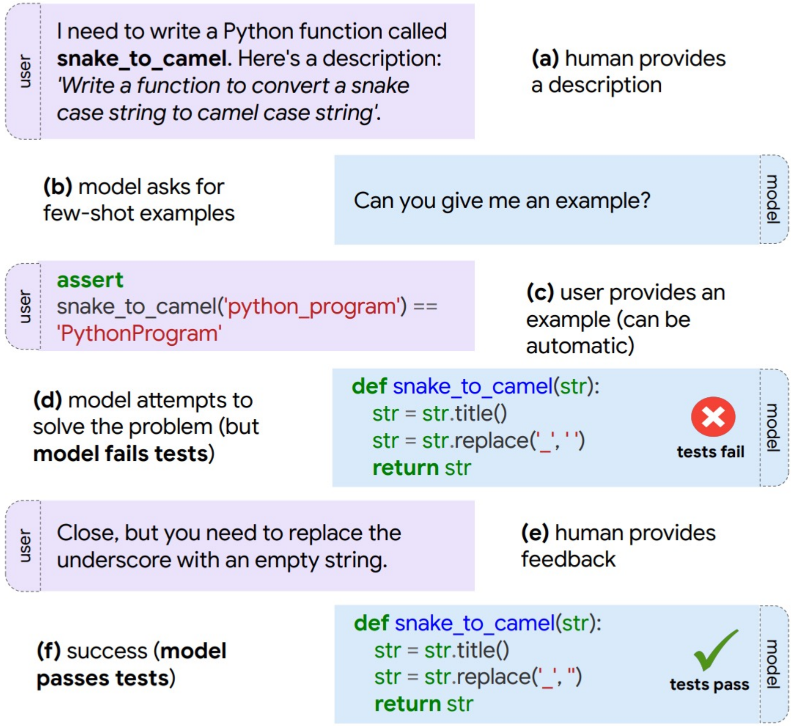


Figure 12: An overview of the "flow" of the human-model collaboration experiments. The human gives a description of the desired program and then guides the model toward the correct solution via dialog.

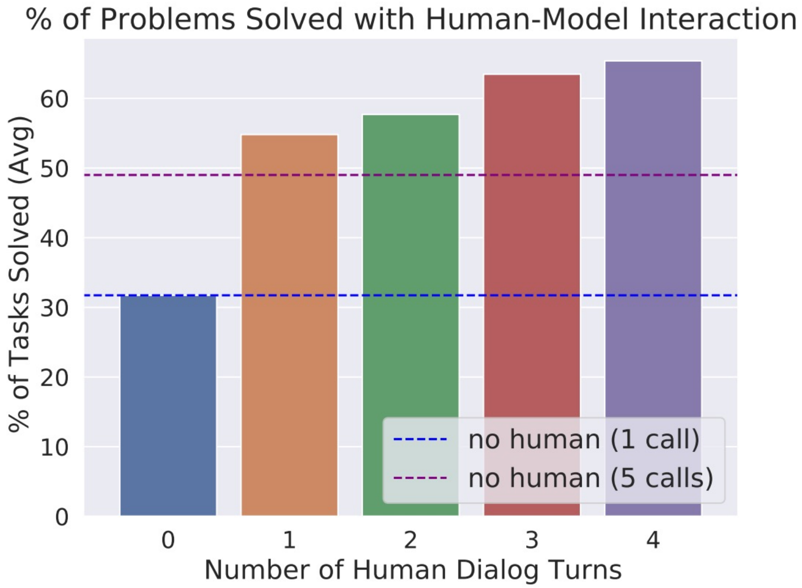
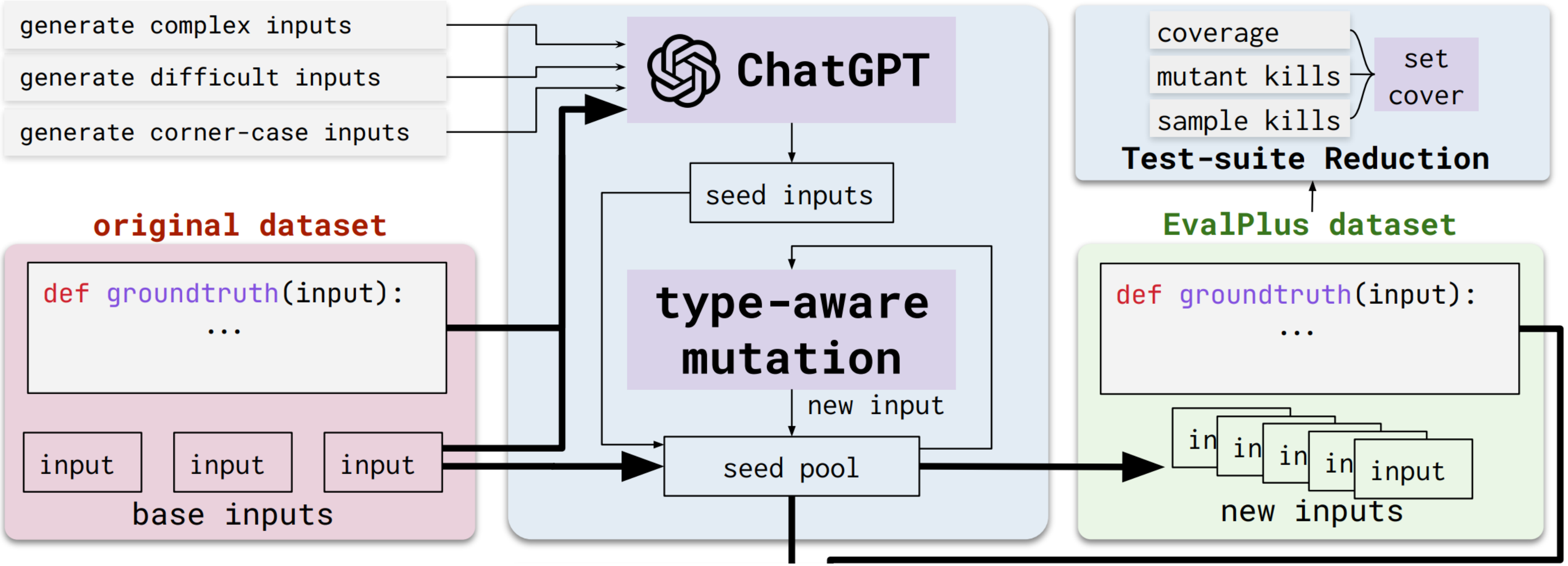


Figure 13: Percent of problems solved as the number of human dialog interventions increases. With 4 interventions, the solve rate increases from 30% to over 65%. Except for the purple horizontal baseline (which corresponds to 5 samples from the model), all pass-rates in this figure were computed using a single sample from the model.

Automated & Improved Testing

- EvalPlus: use LLMs and *fuzzing* (type-aware mutation) to create test cases
- Prompt ChatGPT with the GT solution, some inputs, and instructions to generate more



Liu et al. 2023

Automated & Improved Testing

- EvalPlus: use LLMs and *fuzzing* (type-aware mutation) to create test cases
- Fuzzing: mutate inputs to the functions, apply the groundtruth function, and use the input-output pair to make a new test case.

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)

Liu et al. 2023

Automated & Improved Testing

- EvalPlus: use LLMs and *fuzzing* (type-aware mutation) to create test cases
- Optionally, minifiy the test sets while preserving code coverage and edge case detection.

Table 2: Overview of EvalPlus-improved benchmarks.

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

	Size	pass@k	k=1*	k=1	k=10	k=100
GPT-4 [49]	N/A	base	88.4			
		+extra	76.2			
Phind-CodeLlama [52]	34B	base	71.3	71.6	90.5	96.2
		+extra	67.1	67.0	85.0	92.5
WizardCoder-CodeLlama [38]	34B	base	73.2	61.6	85.2	94.5
		+extra	64.6	54.5	78.6	88.9
ChatGPT [48]	N/A	base	73.2	69.4	88.6	94.0
		+extra	63.4	62.5	82.1	91.1

Liu et al. 2023

Automated & Improved Testing

- EvalPlus: use LLMs and *fuzzing* (type-aware mutation) to create test cases
- These extra tests substantially reduce the pass@k of many models!

	Size	pass@k	k=1*	k=1	k=10	k=100
GPT-4 [49]	N/A	base	88.4			
		+extra	76.2			
Phind-CodeLlama [52]	34B	base	71.3	71.6	90.5	96.2
		+extra	67.1	67.0	85.0	92.5
WizardCoder-CodeLlama [38]	34B	base	73.2	61.6	85.2	94.5
		+extra	64.6	54.5	78.6	88.9
ChatGPT [48]	N/A	base	73.2	69.4	88.6	94.0
		+extra	63.4	62.5	82.1	91.1

Liu et al. 2023

MultiPL-E

- Key idea: it's relatively easy to translate test cases on simple types (e.g. no matrices or functions) from Python to other languages.
- This allows porting HumanEval & MBPP to 18 other languages.

(a) Original Python assertion.

```
assert lsi([0]) == (None, None)
```

(b) Equivalent R.

```
if(!identical(lsi(c(0)), c(NULL, NULL))){
    quit('no', 1)}
```

(c) Equivalent JavaScript.

```
assert.deepEqual(lsi([0]), [void 0, void 0]);
```

Figure 4: Example of a translated assertion.

(a) Original Python docstring from HumanEval #95.

```
Given a dictionary, return True if all keys are
strings in lower case or all keys are strings in
upper case, else return False. The function should
return False is the given dictionary is empty.
```

(b) Terminology translated to Perl.

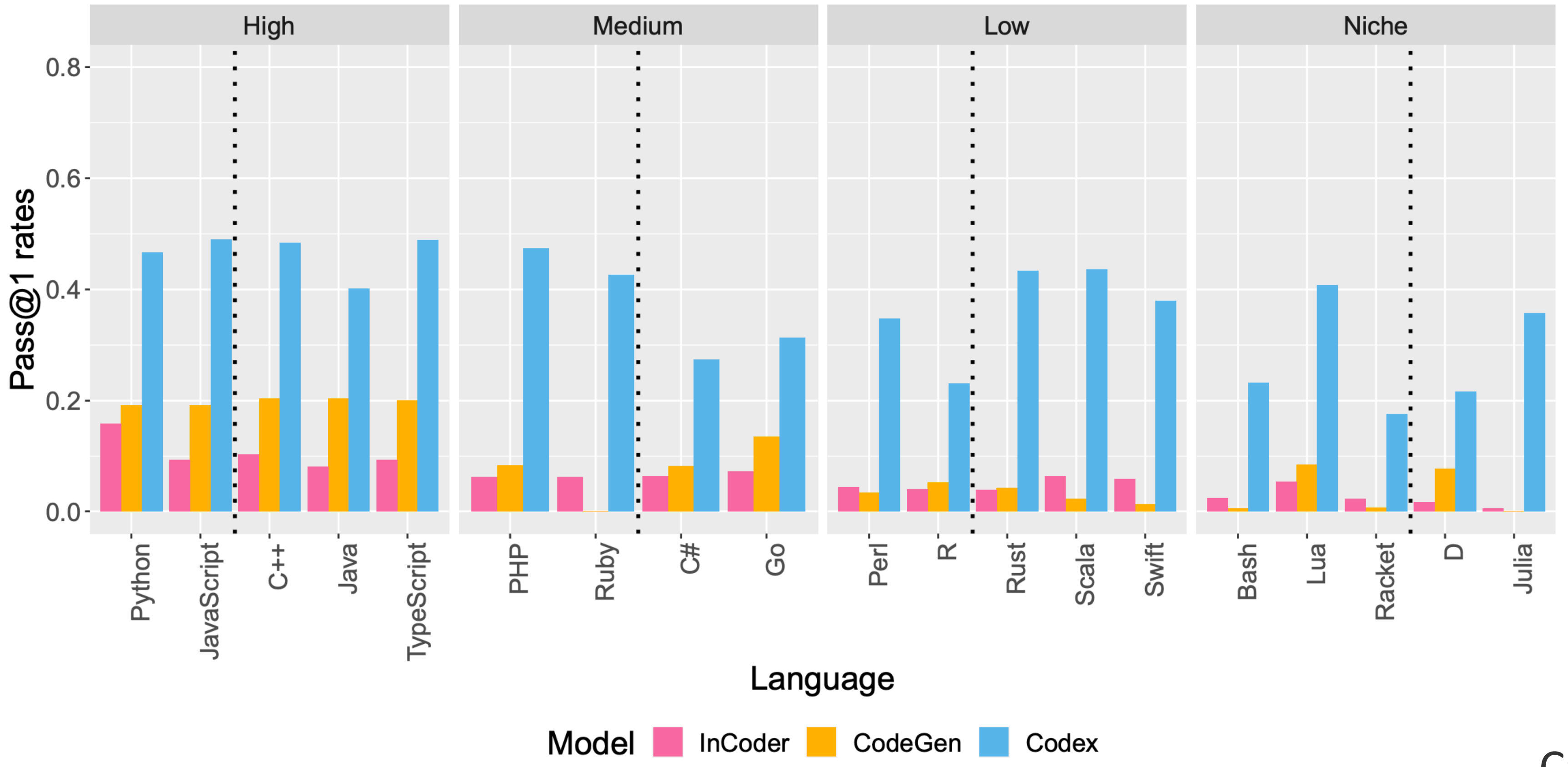
```
Given a hash, return 1 if all keys are strings
in lower case or all keys are strings in upper case,
else return "". The function should return "" is
the given hash is empty.
```

Figure 5: A Python docstring and its Perl translation. Errors (e.g., "is" for "if") are from the original benchmark.

Cassano et al. 2022

MultiPL-E

- Models are generally better on “high-resource” languages with more code on GitHub.
- More analysis of this in the Data lecture, with Starcoder.



Cassano et al. 2022

Incorrect Code Can Be Valuable Too!

- Code might be easily editable to achieve a good solution.

*Lev*enshtein distance: number of character edits required to transform.

$$\textsc{Edit-Sim} = 1 - \frac{lev(gen, ref)}{max(len(gen), len(ref))}$$

Reference Code Snippet

```
def even_odd_count(num):
  even_count = 0
  odd_count = 0
  for i in str(abs(num)):
      if int(i)%2==0:
          even_count +=1
      else:
          odd_count +=1
  return (even_count, odd_count)
```

Generated Code Snippet

```
def even_odd_count(num):
  even_count = 0
  odd_count = 0
  for i in str(num):
      if int(i) % 2 == 0:
          even_count += 1
      else:
          odd_count += 1
  return even_count, odd_count
```

Functional Metric

pass = **0**

Similarity Metric

edit similarity = **0.93**

Human preference

preference = **0.9**

Dibia et al. 2022

Incorrect Code Can Be Valuable Too!

- Dibia et al. compare metrics to evaluate 5 model outputs on HumanEval.
  - EditDistance, BLEU, Pass@1

- Professional programmers with Python experience rate on:
  - **Accuracy**: judge if the snippets are functionally equivalent (judging is easier than writing!)
  - **Value**: How useful is the snippet as a starting point?
  - **Effort**: how much effort to modify the solution into a correct one?

Dibia et al. 2022

Incorrect Code Can Be Valuable Too!

- Value is nearly perfectly correlated with effort (accuracy less so).
- Of all metrics, Pass@1 is most correlated with Value
- But, Edit sim > BLEU and a combination is best (as dissimilar, incorrect code is bad).

$$\textsc{Combined} = min(1.0, \textsc{Pass} + \textsc{Edit-Sim})$$

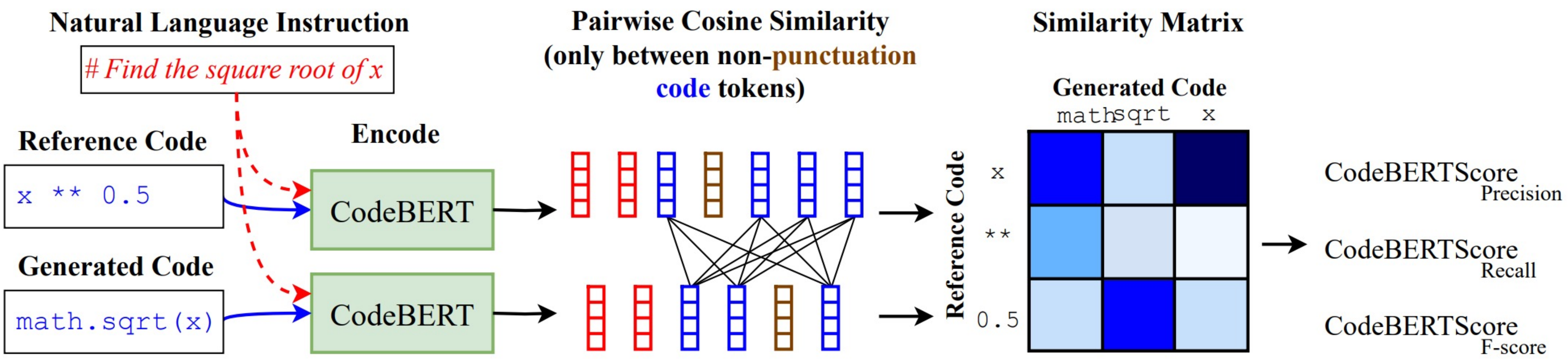
	Human Judgement			Offline Metrics			
	Value	Accuracy	Effort	Pass	Edit Sim	bleu	**Combined**
Value	1.00						
Accuracy	0.87	1.00					
Effort	0.94	0.86	1.00				
Pass	**0.61**	0.66	0.62	1.00			
Edit Sim	**0.48**	0.46	0.51	0.33	1.00		
bleu	**0.36**	0.34	0.39	0.19	0.68	1.00	
Combined	**0.70**	0.71	0.72	0.89	0.61	0.38	1.00

Dibia et al. 2022

CodeBERTScore: Model-based Evaluation

- Captures some intuitions about incorrect code being useful
- BLEU and edit distance only give points for exactly matching code
- Takes NL code descriptions into account

- Use vector similarity from CodeBERT representations
- Recall: every reference vector has >=1 candidate vector with high similarity
- Precision: every candidate vector has >=1 reference vector with high similarity



Zhou et al. 2023

Domains of Code

HumanEval Looks Like Toy Examples?

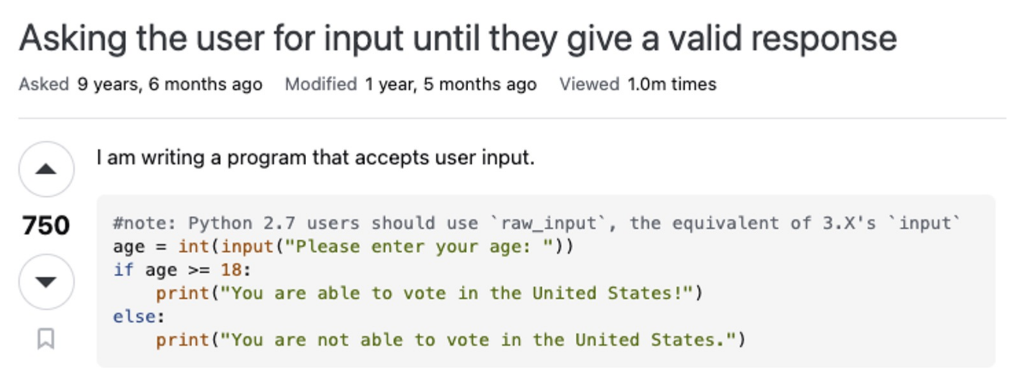
HumanEval Examples

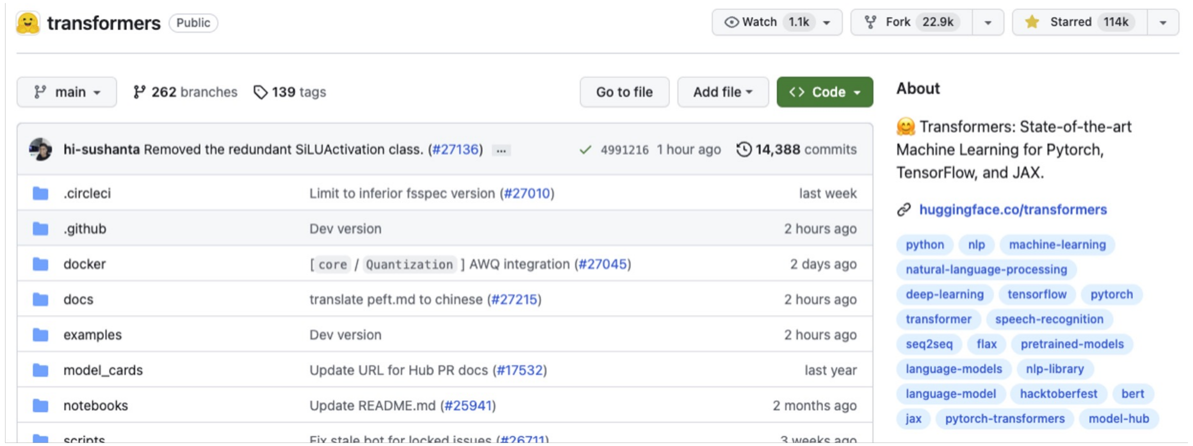
```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```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)
```

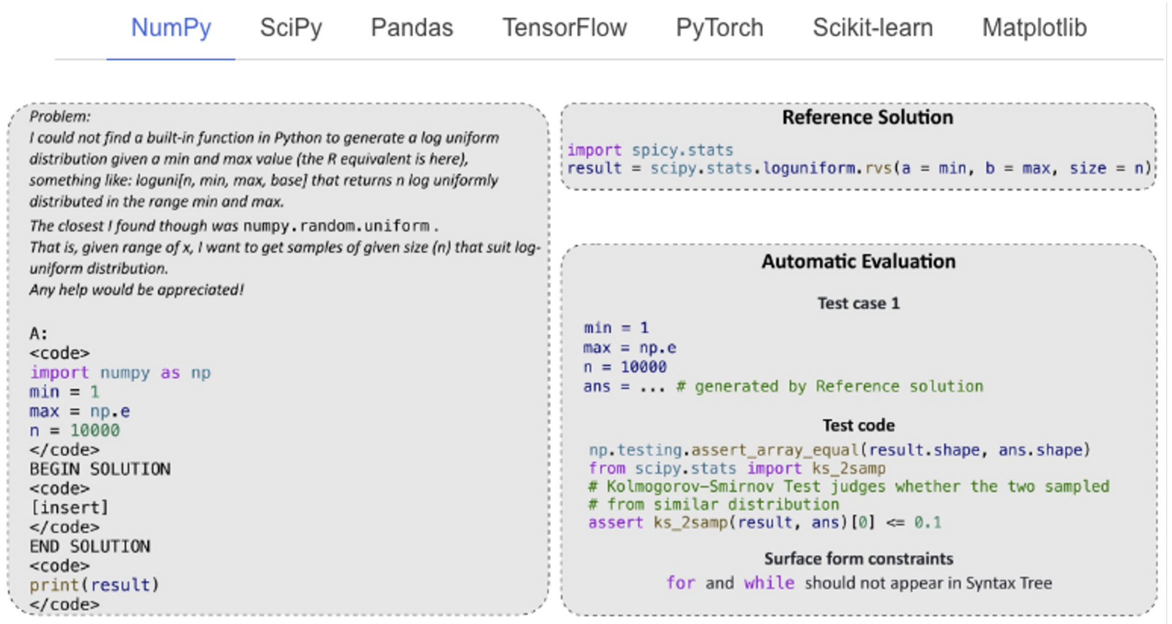
Real-World Development





Broadening Domains

- 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
  - … …
- Open Domain: ODEX
  - 79 Python libraries
  - Four natural languages



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

APPS

- 10,000 problems taken from contest and exercise sites
- 3 difficulty levels
- Has test cases

Problem

You are given two integers n and m. Calculate the number of pairs of arrays (a, b) such that: the length of both arrays is equal to m; each element of each array is an integer between 1 and n (inclusive); $a_i \leq b_i$ for any index i from 1 to m; array a is sorted in non-descending order; array b is sorted in non-ascending order. As the result can be very large, you should print it modulo $10^9 + 7$. Input: The only line contains two integers n and m ($1 \leq n \leq 1000$, $1 \leq m \leq 10$). Output: Print one integer – the number of arrays a and b satisfying the conditions described above modulo $10^9 + 7$.

Model Output

```
n, m = map(int, input().split())
mod = 10**9+7

fact = [1, 1]
factinv = [1, 1]
inv = [0, 1]

for i in range(2, n+1):
    fact.append((fact[-1] * i) % mod)
    inv.append((-inv[mod % i] * (mod//i)) % mod)
    factinv.append((factinv[-1] * inv[-1]) % mod)

def cmb(n, r, mod):
    if (r < 0) or (n < r):
        return 0
    r = min(r, n - r)
    return fact[n] * factinv[r] * factinv[n-r] % mod

print((cmb(n, m, mod) * cmb(m-1, n-1, mod)) % mod)
```

CodeContests (AlphaCode)

- Similar to APPS but focuses on improving test coverage, via input mutation (like EvalPlus).

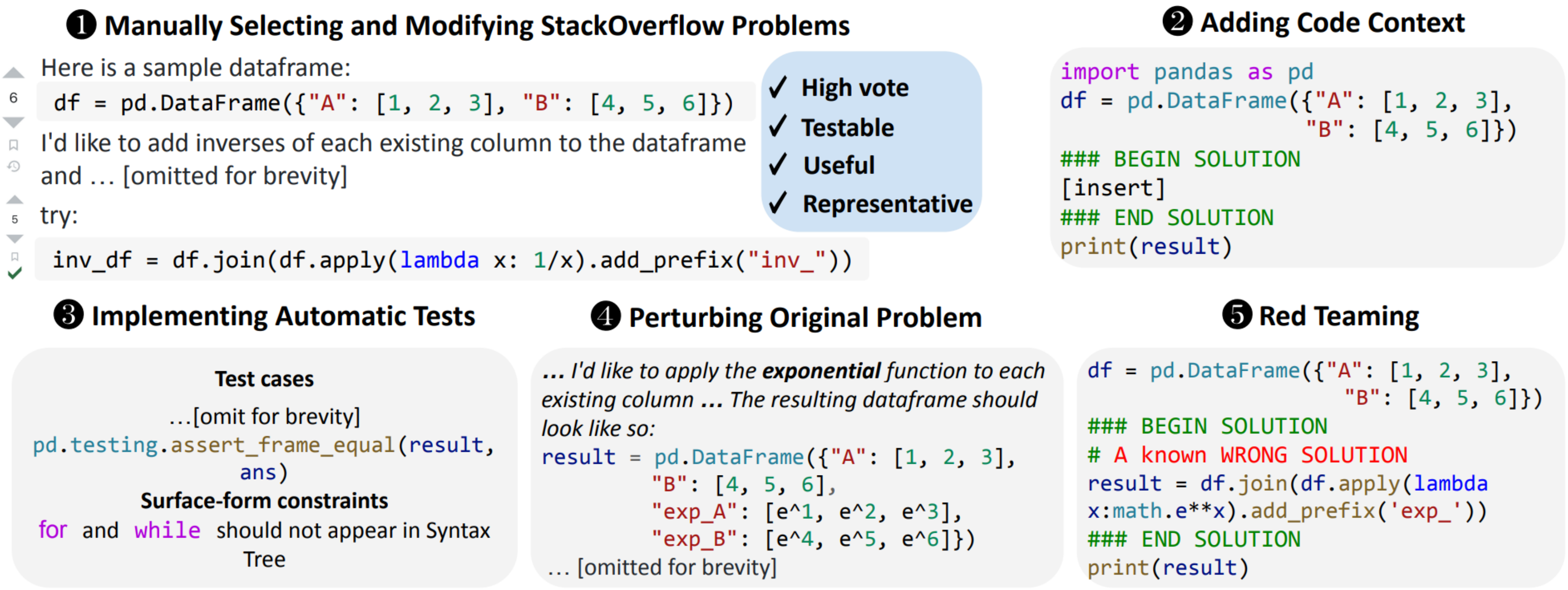
		Tests per problem			Solutions per problem (% correct)		
Split	Problems	Example	Hidden	Generated	C++	Python	Java
Train	13328	2.0	14.8	79.1	493.4 (27%)	281.1 (47%)	147.9 (46%)
Valid	117	1.5	12.9	190.0	231.6 (47%)	137.2 (55%)	131.1 (54%)
Test	165	1.7	9.4	192.7	196.0 (45%)	97.3 (54%)	105.2 (51%)

- Manual inspection shows high false-positive rate of model-produced solutions.

Dataset	Tests / problem	False Positive (FP) Rate	FP or Slow Rate
APPS	20.99	60%	70%
HumanEval	7.77	30%	N/A
CodeContests raw	12.4	62%	88%
CodeContests	**203.7**	**4%**	**46%**

DS-1000

- 1,000 data science problems, based on StackOverflow questions
- Domain-specific test cases, e.g. matplotlib plots have their elements programmatically extracted



DS-1000

- Perturb the problems to reduce chances of memorization, since models may have been trained on StackOverflow
- ”Surface” perturbations: don’t change solution. “Semantic”: do, but try to keep difficulty the same (e.g. max -> min)

	Pandas	NumPy	Scikit-learn	SciPy	TensorFlow	PyTorch	Overall
$\text{Origin}_{\text{surface}}$	37.3	61.2	52.6	33.0	64.9	64.8	53.2
Surface	31.9 −5.4	58.4 −2.8	55.7 +3.1	32.1 −0.9	58.0 −8.9	50.0 −14.8	49.8 −3.4
$\text{Origin}_{\text{semantic}}$	36.8	56.7	60.6*	40.3	71.3	65.1	47.2
Semantic	33.2 −3.6	49.0 −7.7	38.9* −21.7	34.3 −6.0	42.5 −25.8	30.5 −34.6	38.2 −9.0
$\text{Origin}_{\text{difficult}}$	39.9	52.7	5.0*	58.1	73.0*	53.8*	46.8
Difficult Rewrite	17.7 −22.2	27.1 −25.6	0.0* −5.0	13.8 −44.3	38.0* −35.0	28.8* −25.0	21.0 −25.8

ARCADE

- Executable problems from Jupyter notebooks

Prompt Prefix (Exemplars)

```
Which countries host at least two Olympic games?
# Solution: Let's solve this problem step-by-step.     preamble
# Step 1: Get the counts each country hosted Olympics
count_df = df['Country'].value_counts()
# Step 2: Get the rows whose average score is above 90
filtered_df = count_df[count_df >= 2]
# Step 3: Get the country names as a list              explanation
filtered_df.index.tolist()
```

```
[1] import pandas as pd
    df = pd.read_csv('stores.csv')
```

```
[2] # Schema of Dataframes:
    # Columns in df with example values:
    # Stu_Name (Mike),Engineering (90),English (89),Math (92)
```

```
[3] Get the students with an average score above 90
    for science subjects
```

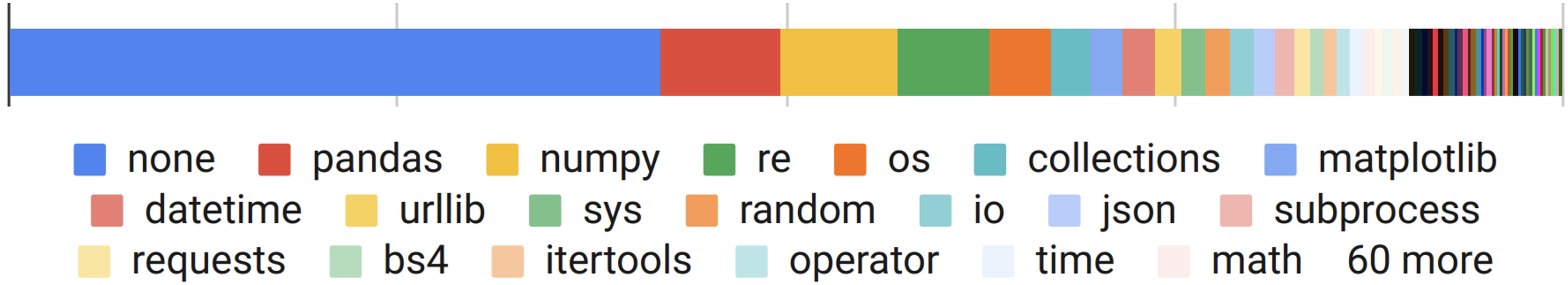
```
[3a] ▸Vanilla Prediction (no exemplars):
     df['Science_Avg'] = (df['Engineering']+df['Math'])/2
     df[df['Science_Avg'] > 90][['Stu_Name','Science_Avg']]
```

Models	*pass*@30	# API	Lines of Code (LoC)	Comment Lines	Tokens / Line	API / Line
Baseline (Tab. 2)	47.7	4.9	2.3	0.1	21.1	3.2
+ More Context	49.3	4.9	2.3	0	21.1	3.1
Prompting with Additional Few-shot Exemplars						
Vanilla Code	49.9	5.3	2.4	0.1	20.8	3.1
Step-by-Step Code	51.9	5.6	3.2	0.1	17.8	2.7
+ Preamble	51.9	5.9	3.5	0.2	16.9	2.5
+ Pre. + Explanation	52.5	6.8	4.2	3.3	14.9	2.2

```
[3b] ▸Step-by-Step Prompting (with exemplars):
     # Solution: Let's solve this problem step-by-step.     preamble
     # Step 1: Create a new column with the average score of
     # engineering and math                                  explanation
     df['Science_Avg'] = (df['Engineering'] + df['Math']) / 2
     # Step 2: Get the rows whose average score is above 90
     df_score_above_90 = df[df['Science_Avg'] > 90]
     # Step 3: Return the student name and average scores
     result = df_score_above_90[['Stu_Name', 'Science_Avg']]
```

ODEX: Open-Domain, with Evaluation

- 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)
```

a

```
    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'
```

a

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'
```

d

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

random.randint(3, 5) → 3, 4, 5

- Specialized verification
- (Potentially) not reproducible queries
  - HTTP requests, e.g., `requests.post(“https://def.xyz”, data={‘key’: ‘value’})`

```
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
```

Significant Performance Gaps: Open vs. Closed

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

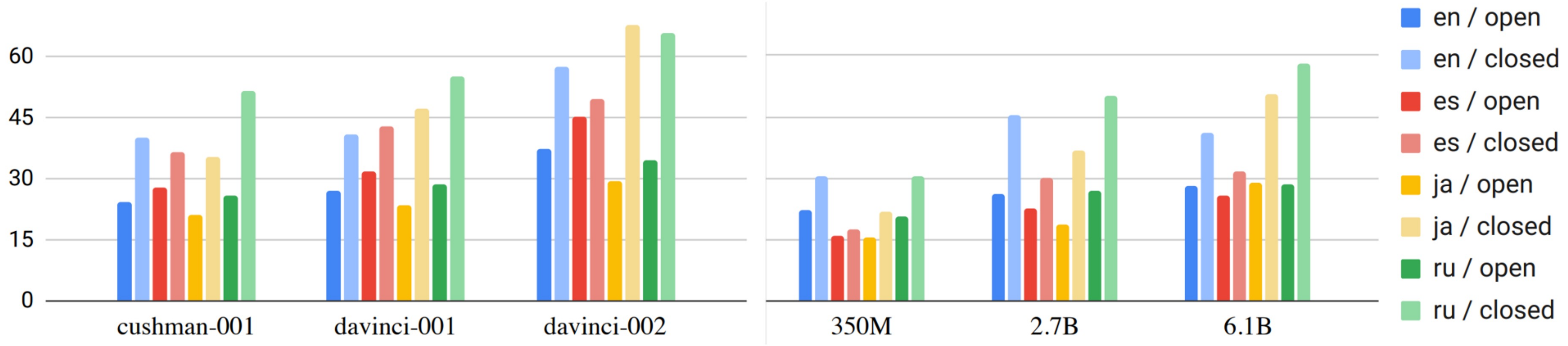


Figure 7: CODEX (left) and CODEGEN (right) *pass@1* on open- and closed-domain problems in each language.

Code Complexity

Functional C

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

```
from datetime import datetime                                   Import Statements
class VendingMachine:                                           Class Name
    """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. """
                                                                Class Description
  def __init__(self):
    """
    Initializes the vending machine's inventory and balance.
    """
                                                                Class Constructor
    self.inventory= []
    self. balance= {}
  def purchase_item(self, item_name):                           Method Signature
    """ Purchases a product from the vending machine and returns the balance after the
purchase.                                                       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:                                  Test Fixtures:
    self.vm = VendingMachine()                                  setUp
    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.                                      Parameter/Return Description
    >>> vendingMachine.inventory = {'Coke': {'price': 1.25, 'quantity': 10}}
    >>> vendingMachine.restock_item('Coke', 10)
    True
    >>> vendingMachine.inventory
    {'Coke': {'price': 1.25, 'quantity': 20}}
    """                                             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

Retrieved Code

```
# 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:"""
```

Unfinished Code

```
@unittest.skipIf(torch_device != "cuda")
def test_float16_inference(self):
    components = self.get_dummy_components()
```

Model Prediction

```
    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.