InCoder, SantaCoder, and StarCoder: Findings from Training Code LLMs

Daniel Fried, with many others
from Meta AI and the BigCode project

Are bigger models the solution for AI-assisted programming?

Posing This Question in 2012…

On the Naturalness of Software

Abram Hindle, Earl Barr, Mark Gabel, Zhendong Su, Prem Devanbu

[ICSE 2012; Most Influential Paper 2022]

Natural languages like English are rich, complex, and powerful. We begin with the conjecture that most software is also natural, in the sense that it is created by humans at work, with all the attendant constraints and limitations—and thus, like natural language, it is also likely to be repetitive and predictable. We then proceed to ask whether a) code can be usefully modeled by statistical language models and b) such models can be leveraged to support software engineers.

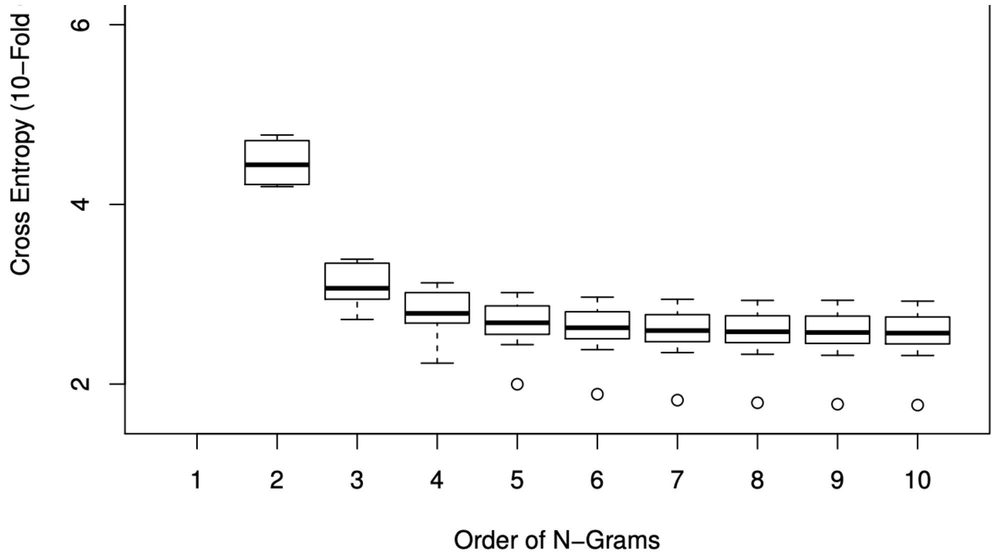
Posing This Question in 2012…

On the Naturalness of Software

Abram Hindle, Earl Barr, Mark Gabel, Zhendong Su, Prem Devanbu

[ICSE 2012; Most Influential Paper 2022]

- n-gram models trained on ~25 million lines of code
- Substantial improvements to Eclipse’s auto-complete
- But, 3-4 orders of magnitude less data than modern neural models

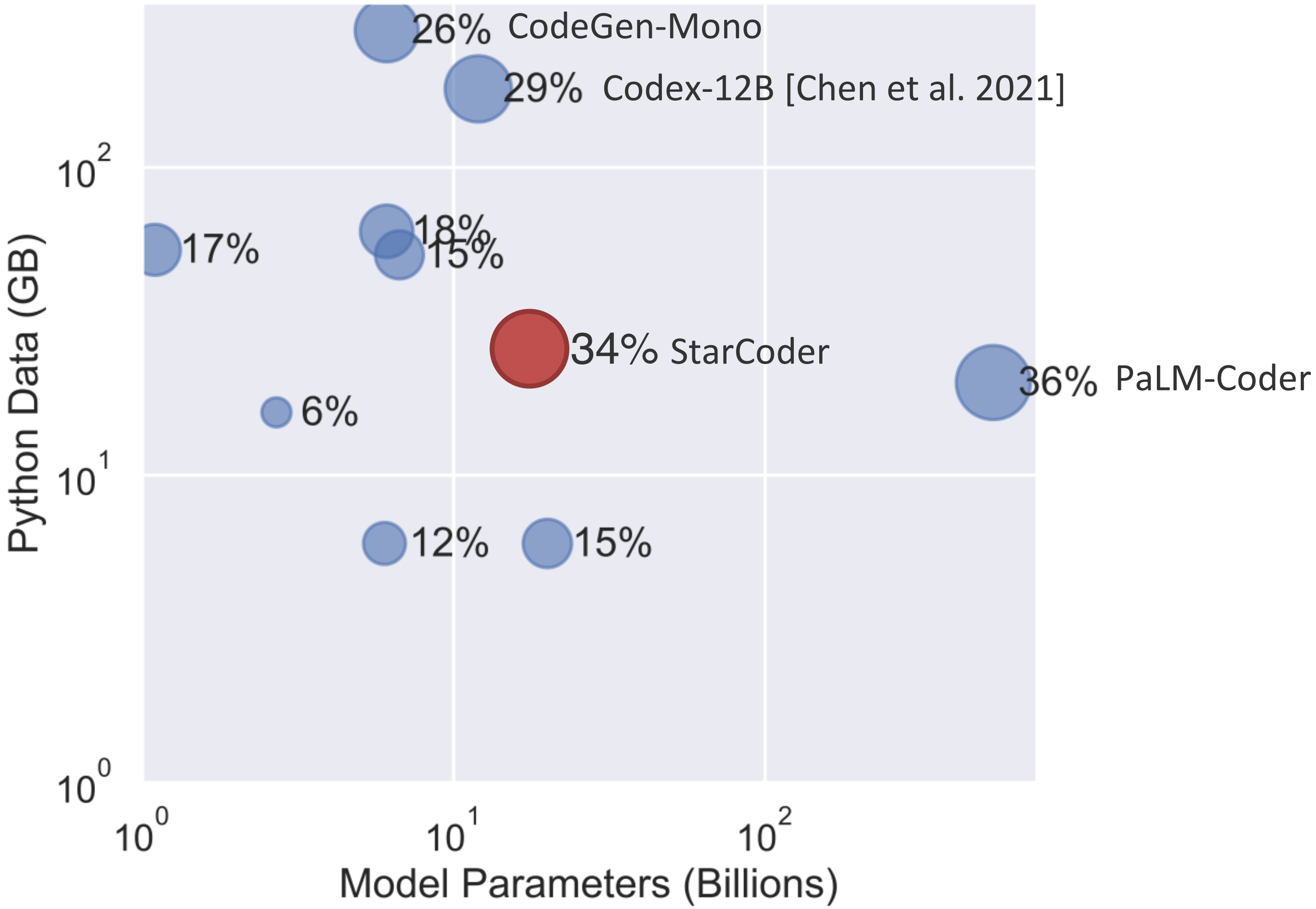


… and now


AI pair programming is here.
75% more fulfilled
55%
faster coding
runtime.go
46%
code written
Now available for
all businesses
Keep flying with your favorite editor
JET
BRAINS

… and now

Function pass rate on a Python docstring-to-function task [HumanEval, Chen et al. 2021]
by amount of Python data & model scale:



[Compiled from Chen et al. 2021, Xu et al. 2021, Li et al. 2021, Fried et al. 2022, Nijkamp et al. 2022, Chowdhery et al. 2022, Li et al. 2023] *part of*

Are bigger models the solution
for AI-assisted programming?

YES

Outline

- InCoder
  - Infilling and natural language data
- The Stack & SantaCoder
  - Data filtering and model improvement experiments
- StarCoder
  - More data: more languages, issues, commits, Jupyter…
  - ***Scale***

Outline

- InCoder
  - Infilling and natural language data
- The Stack & SantaCoder
  - Data filtering and model improvement experiments
- StarCoder
  - More data: more languages, issues, commits, Jupyter…
  - ***Scale***

Meta

LLM Training Objectives

```
def minimize_in_graph(build_loss_fn, num_steps=200, optimizer=None):
```

Prefix

```
  """ Minimize a loss function using gradient.
  Args:
     build_loss_fn: a function that returns a loss tensor for a mini-batch of examples.
     num_steps: number of gradient descent steps to perform.
     optimizer: an optimizer to use when minimizing the loss function. If None, will use Adam
  """
```

Target

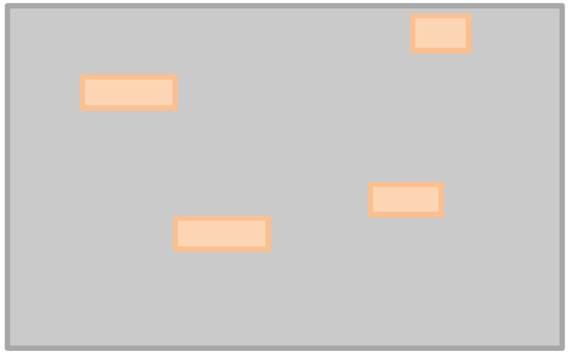
```
  optimizer = tf.compat.v1.train.AdamOptimizer(0.1) if optimizer is None else optimizer
  minimize_op = tf.compat.v1.while_loop(
     cond=lambda step: step < num_steps,
     body=train_loop_body,
     loop_vars=[tf.constant(0)], return_same_structure=True)[0]
  return minimize_op
```

Suffix

“Causal” (L-to-R)

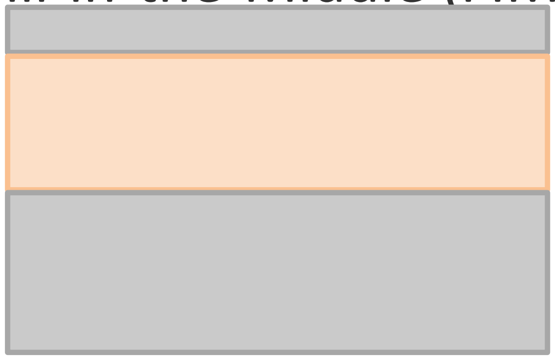
[e.g. GPT-*, Codex]

Masked Infilling



[e.g. BERT, CodeBERT]

“Causal Masking” / Fill-in-the-Middle (FIM)



[Donahue+ 2020, Aghajanyan+ 2022, ours, Bavarian+ 2022]

Causal Masking / FIM Objective

Training

Original Document

```python
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```

[Donahue et al. 2020, Aghajanyan et al. 2022, Fried et al. 2022, Bavarian et al. 2022]

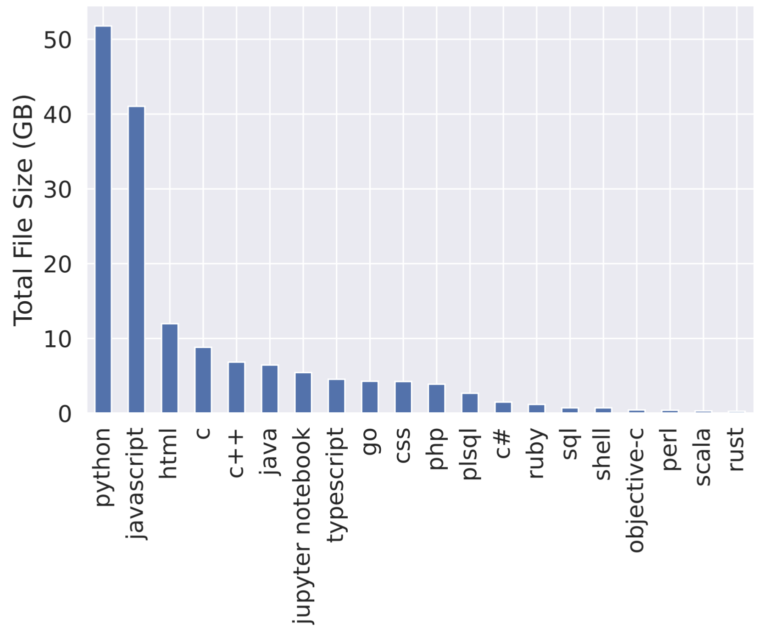
Evaluation: HumanEval Benchmark

Constructed by authors of Codex paper; programming puzzle/simple contest problems. Evaluated using unit tests.

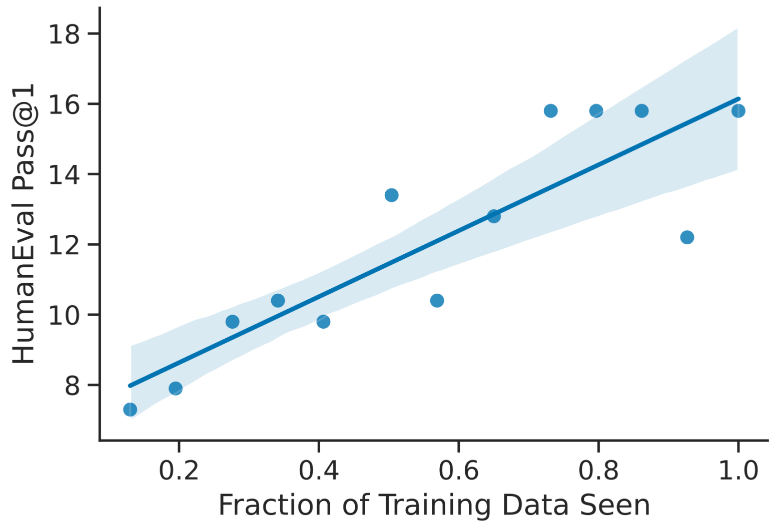
```python
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
        than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```

Model Training

- Training Data
  - 600K permissively-licensed repositories from GitHub & GitLab. ~150GB total
  - StackOverflow: questions, answers, comments. ~50GB



- Models
  - Standard transformer LM
  - 1B model: ~1 week on 128 V100s
  - 6B model: ~3 weeks on 240 V100s



Zero-Shot Software Tasks via Infilling

Zero-shot Inference

Docstring Generation

```python
def count_words(filename: str) -> Dict[str, int]:
    """
    Counts the number of occurrences of each word in the given file.

    :param filename: The name of the file to count.
    :return: A dictionary mapping words to the number of occurrences.
    """
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
        return word_counts
```

Multi-Region Infilling

```python
from collections import Counter

def word_count(file_name):
    """Count the number of occurrences of each word in the file."""
    words = []
    with open(file_name) as file:
        for line in file:
            words.append(line.strip())
    return Counter(words)
```

Evaluation

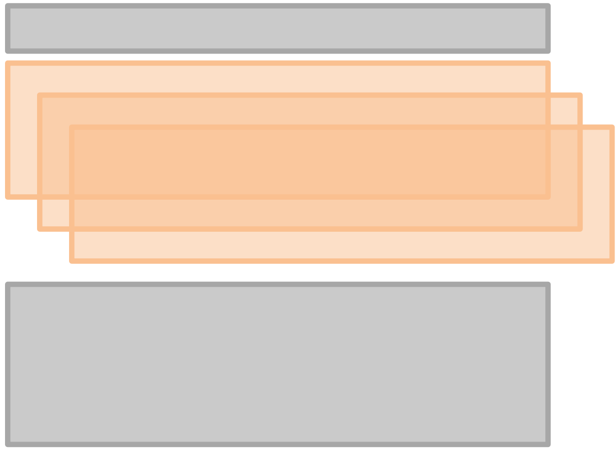
- Zero-shot evaluation on several software development-inspired code infilling tasks (we'll show two).
- Compare the model in three different modes to evaluate benefits of suffix context

Baselines

Left-to-Right Single

Doesn't use suffix

Left-to-Right Rerank



Only uses suffix when reranking

Ours

Causal Masking

Uses suffix when generating

Evaluation: Function Completion

Fill in one or more lines of a function; evaluate with unit tests.

```python
from typing import List

def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """
    Check if in given list of numbers, are any two numbers closer to each other
        than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True
    return False
```

Method	Pass Rate	Exact Match
L-R single	24.9	15.8
L-R reranking	28.2	17.6
CM infilling	38.6	20.6

Constructed from HumanEval [Chen et al. 2021]

Evaluation: Docstring Generation

```python
def count_words(filename: str) -> Dict[str, int]:
    """
    Counts the number of occurrences of each word in the given file.

    :param filename: The name of the file to count.
    :return: A dictionary mapping words to the number of occurrences.
    """
    with open(filename, 'r') as f:
            word_counts = {}
            for line in f:
                for word in line.split():
                    if word in word_counts:
                        word_counts[word] += 1
                    else:
                        word_counts[word] = 1
        return word_counts
```

Method	BLEU
Ours: L-R single	16.05
Ours: L-R reranking	17.14
Ours: Causal-masked infilling	18.27

[CodeXGlue, Lu et al. 2021]

Evaluation: Return Type Prediction

Type Inference

```python
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```

Method	F1
Ours: Left-to-right single	30.8
Ours: Left-to-right reranking	33.3
Ours: Causal-masked infilling	**59.2**
TypeWriter (Supervised)	48.3

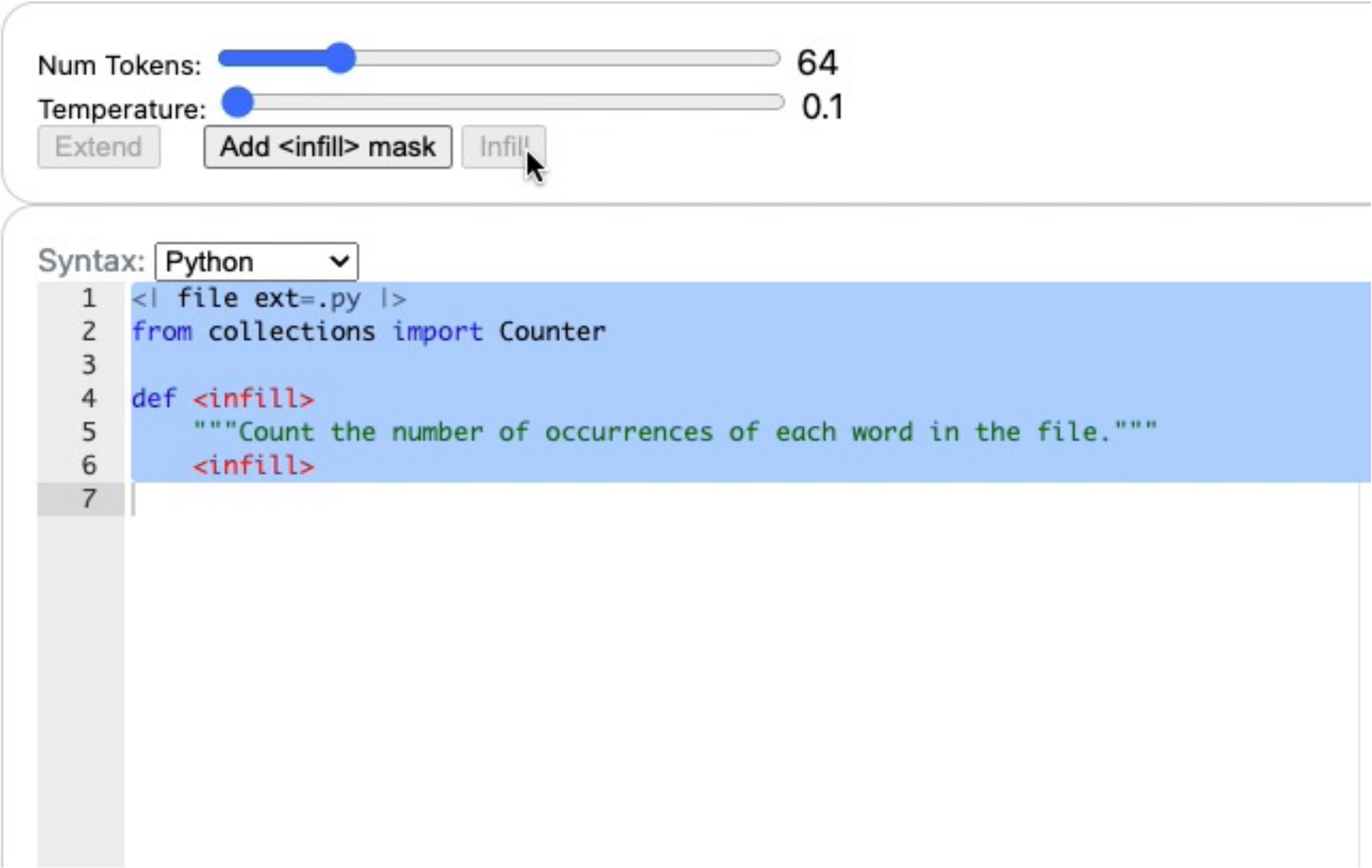
[TypeWriter OSS, Pradel et al. 2020]

Ablations

- **StackOverflow data improves performance**
- **Comparable performance from infilling and non-infilling models**

#	Size (B)	Obj.	Training Data	Data Size	Train Tokens	HumanEval Pass@1	MBPP Pass@1
1)	6.7	CM	multi lang + SO	204 GB	52 B	15	19.4
2)	1.3	CM	multi lang + SO	204 GB	52 B	8	10.9
3)	1.3	LM	multi lang + SO	204 GB	52 B	6	8.9
4)	1.3	LM	Python + SO	104 GB	25 B	9	9.8
5)	1.3	LM	Python	49 GB	11 B	5	6.1

Demo



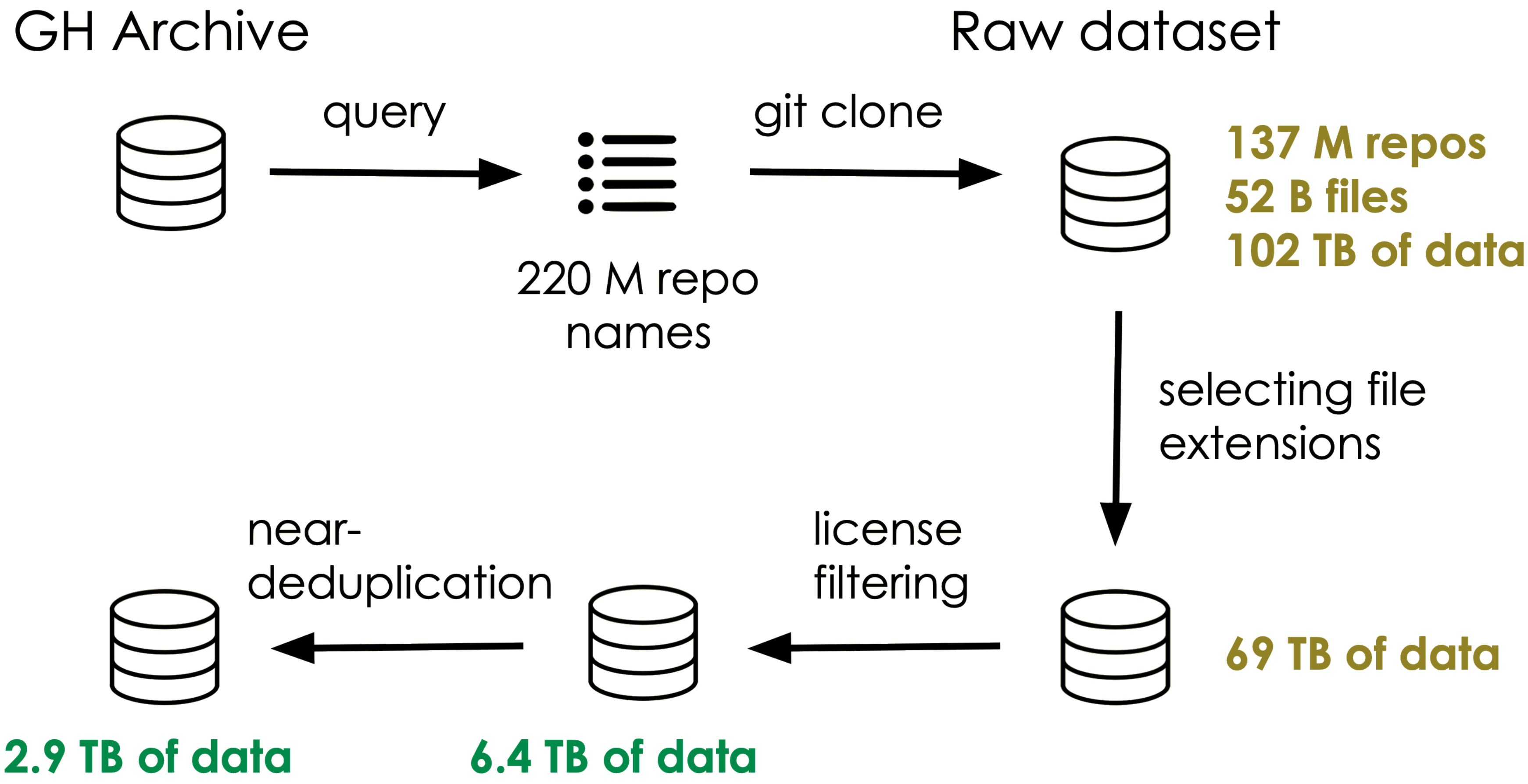
Demo: huggingface.co/spaces/facebook/incoder-demo

Outline

- InCoder
  - Infilling and natural language data
- The Stack & SantaCoder
  - Experiments with data filtering and model improvements
- StarCoder
  - More data: more languages, issues, commits, Jupyter…
  - *Scale*

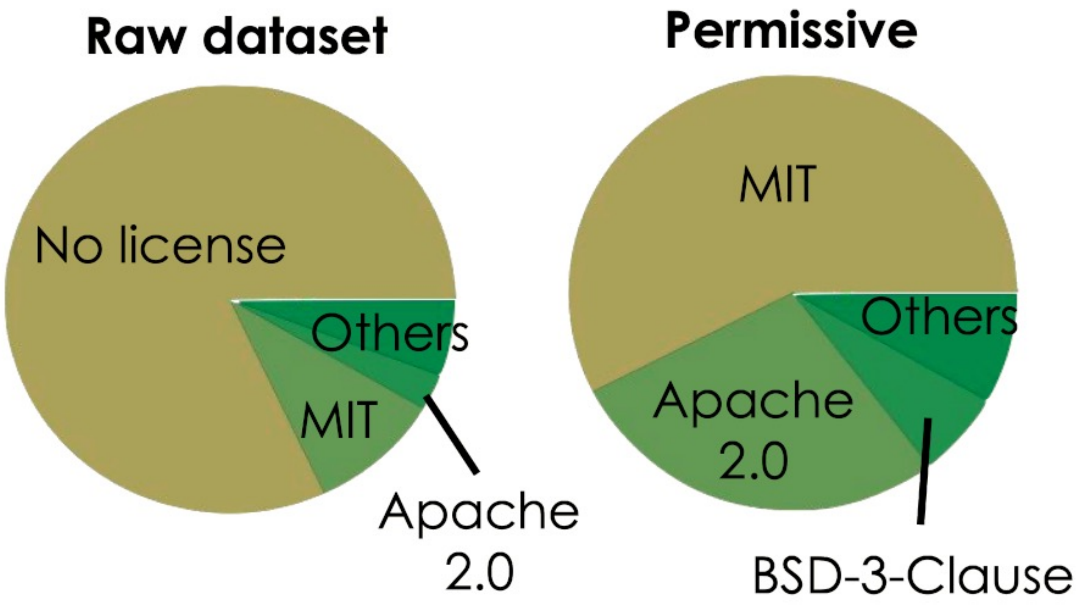
BigCode

The Stack: Dataset



[Kocetkov et al. 2022]

The Stack: Dataset



Permissive license distribution of licenses used to filter the dataset:

MIT (67.7%) | Apache-2.0 (19.1%) | BSD-3-Clause (3.9%) | Unlicense (2.0%) | CC0-1.0 (1.5%) | BSD-2-Clause (1.2%) | CC-BY-4.0 (1.1%) | CC-BY-3.0 (0.7%) | 0BSD (0.4%) | RSA-MD (0.3%) | WTFPL (0.2%) | MIT-0 (0.2%) | Others (166) (2.2%)

[Kocetkov et al. 2022]

The Stack: Python Models

- Possible to approximate Codex-12B performance with permissively licensed data? Train 350M models on Python
- ***Deduplication always improves performance*** (https://huggingface.co/blog/dedup)
- License filtering hurts, but there's enough data available to match Chen et al. 2021

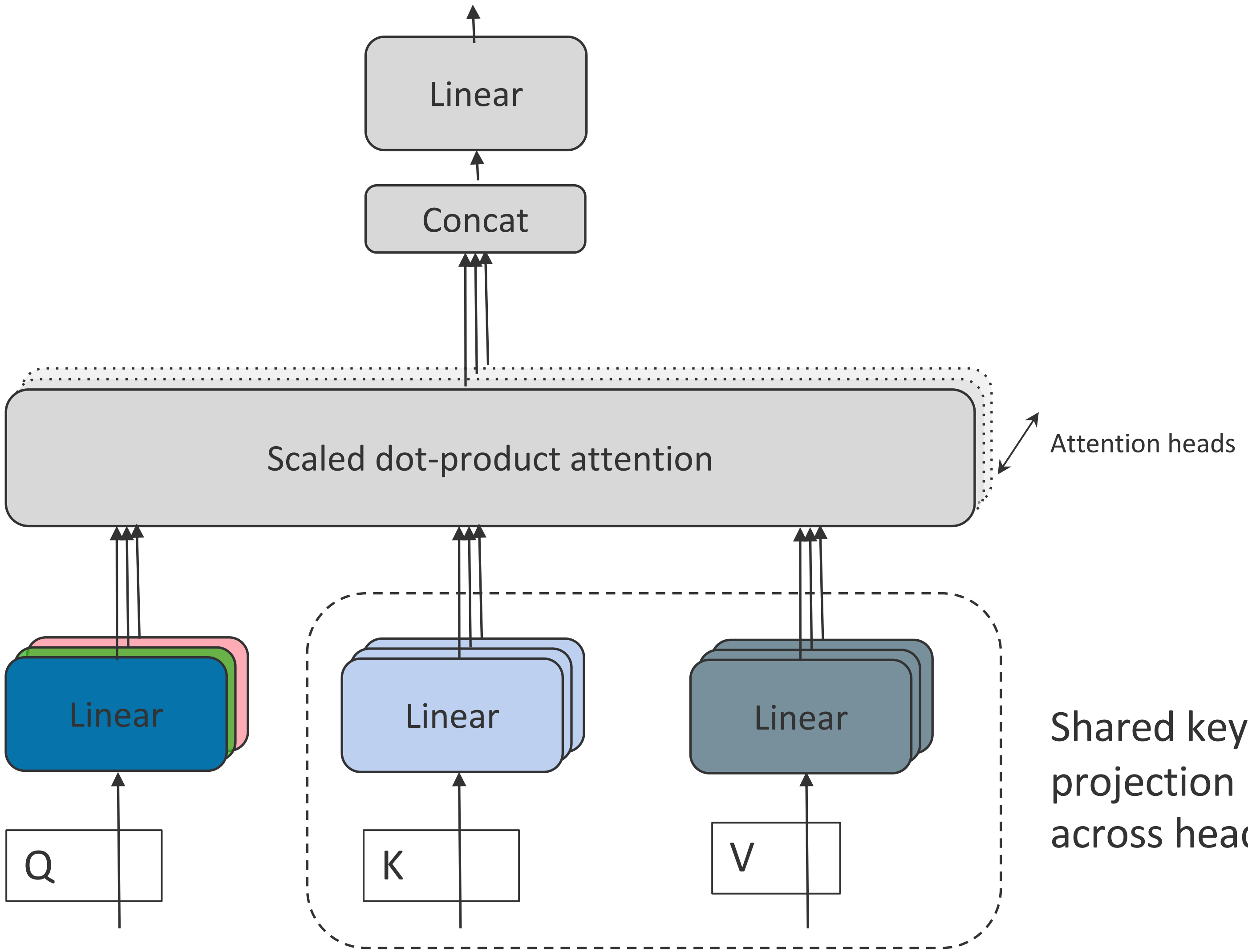
Dataset	Filtering	pass@1	pass@10	pass@100	Python Data
Codex (300M)	Exact-dedup?	13.17	20.17	36.27	180 GB
CodeGen (350M)	unknown	12.76	23.11	35.19	
Python all-license	None	13.11	21.77	36.67	740 GB
	Near-dedup	**17.34**	**27.64**	**45.52**	
Python permissive-license	None	10.99	15.94	27.21	191 GB
	Near-dedup	**12.89**	**22.26**	**36.01**	80 GB

SantaCoder: Overview

- Preparation for a big run: explorations at 1B scale
- Data: The Stack
- Tokenizer: BPE following GPT-2 recipe; use a digit splitter
- Ablations
  - Multi-query attention and infilling (FIM, Bavarian et al. 2022)
  - Data filtering

Multi-Query Attention

- Designed to reduce memory-bandwidth cost to speed up inference



Shared key, value projection parameters across heads

Shazeer, 2019

SantaCoder: Model Ablations

- Infilling (FIM) and MQA “for cheap”

Language	Attention	FIM	HumanEval	MBPP
Java	Multi Query Attention	✓	0.35	0.54
	Multi Head Attention	✓	0.36	0.55
	Multi Query Attention	✗	0.37	0.55
JavaScript	Multi Query Attention	✓	0.33	0.64
	Multi Head Attention	✓	0.37	0.67
	Multi Query Attention	✗	0.37	0.65
Python	Multi Query Attention	✓	0.36	0.67
	Multi Head Attention	✓	0.38	0.70
	Multi Query Attention	✗	0.39	0.68

Table 5: Pass@100 results for the architecture ablations on HumanEval and MBPP.

SantaCoder: Data Filtering Ablations

- Remove repos with < 5 stars

  - Hurts substantially!

- Remove files with low (or very high) comment-to-code ratio

  ~ Mixed effects

- More aggressive near-duplicate filtering

  + Very slight improvements

- Remove files with low character-to-token ratios

  + Very slight improvements

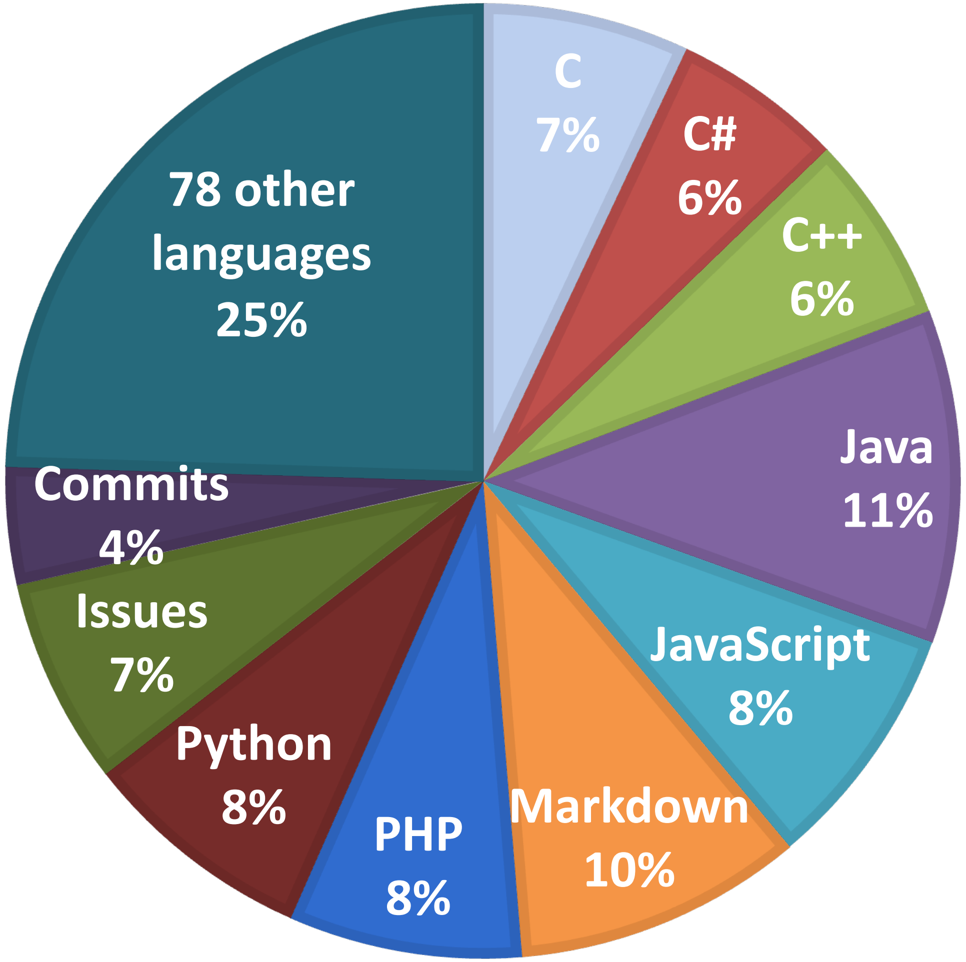
Outline

- InCoder
  - Infilling and natural language data
- The Stack & SantaCoder
  - Experiments with data filtering
- StarCoder
  - More data: more languages, issues, commits, Jupyter…
  - ***Scale***

BigCode

A Large-Scale Multilingual Model

We follow the natural distribution and sample data from 86 languages proportionally to their volume. **800GB total.** Lots of natural language (~20%)!



GitHub Data

Issues (discussion threads)

```
<issue_start>title + USERID: comment
<issue_comment>USERID: comment ...
<issue_closed> <eos>
```

Model input:

```
<issue_start><issue_comment>username_0: I am trying to train a
    text classifier based on BERT using ‘transformers‘ but I
    get the error: ‘RuntimeError: CUDA Out of memory‘.<
    issue_comment>
```

Model output:

```
username_1: Hi,

This is not an issue with transformers. You should look at your
     GPU usage and see if you can free up some space before
    training starts (or use smaller batch sizes).
```

Commits and Commit Messages

Model input:

```
<commit_before>def fibonacci(n):<commit_msg>add type hints to
    function<commit_after>def
```

Model output:

```
def fibonacci(n: int) -> list[int]:
```

Jupyter Notebooks

Model input:

```
<jupyter_text>Let's test our ‘is_prime‘ function:<jupyter_code>
    print(is_prime(3))
print(is_prime(4))
print(is_prime(29))
print(is_prime(33))<jupyter_output>
```

Model output:

```
True
False
True
False
```

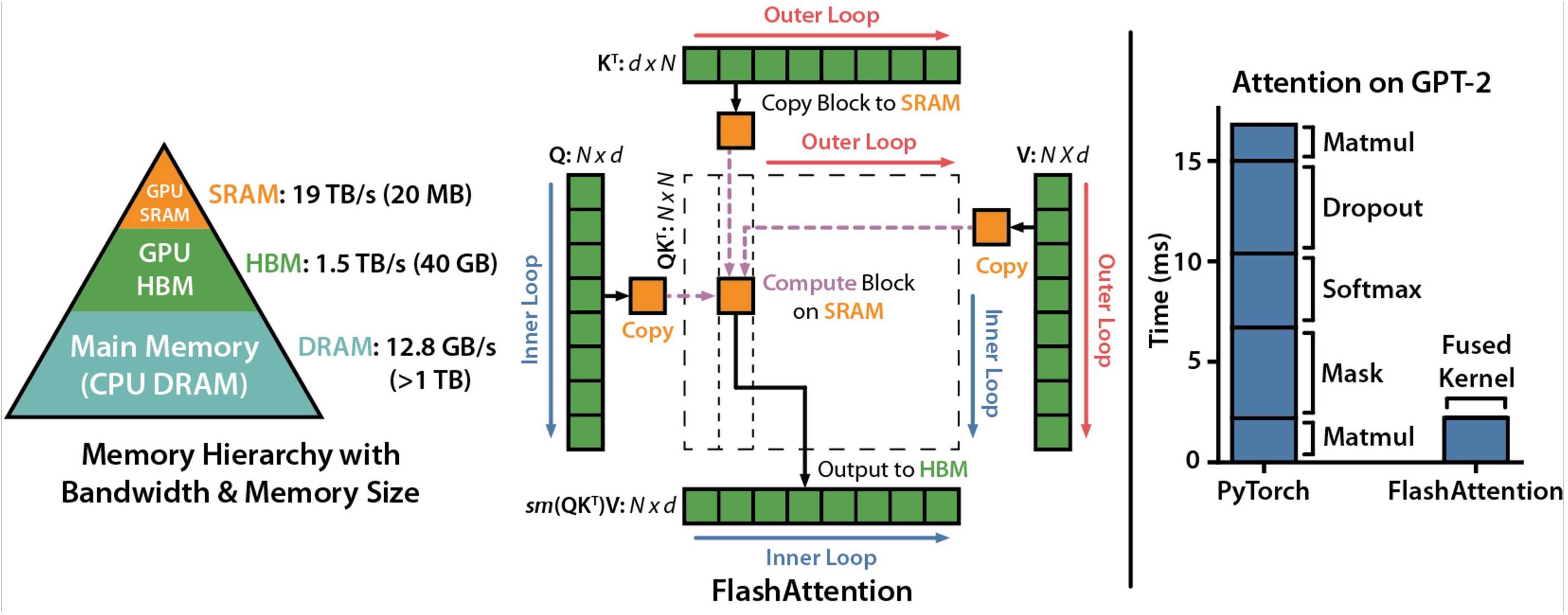
Model input:

```
<jupyter_code>numbers = [1, 9, 8, 3, 27]
print([n*2 for n in numbers])<jupyter_output>
```

Model output:

```
[2, 18, 16, 6, 54]
```

Flash Attention



→ up to 4x speedup over standard attention
→ scale sequence length up to 8192 tokens.

[Dao et al. 2022]

Models

- StarCoderBase
  - 15.5B parameters, trained on 1T tokens (~3 epochs)
    - This is much smaller than Chinchilla optimal, but we were aiming for inference efficiency
    - Multiple epochs didn’t seem to hurt
  - ~1 month on 512 80GB A100s
  - Megatron-LM with BF16 and FlashAttention
- StarCoder
  - Continued training on 35B tokens of Python (two epochs)

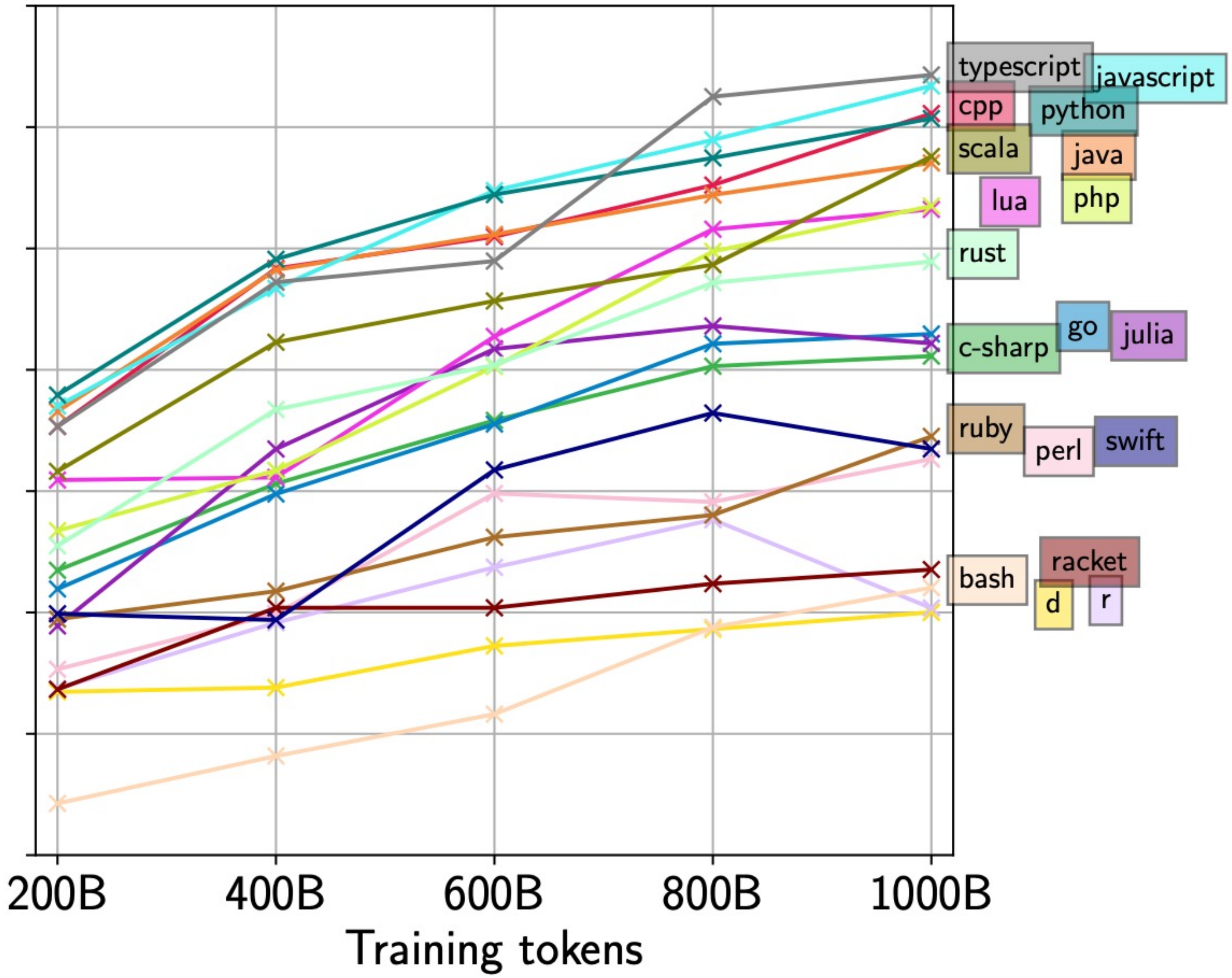
MultiPL-E

- Translations of the HumanEval benchmark into other programming languages.
- Together, StarCoderBase and StarCoder outperform OpenAI’s code-cushman-001 on HumanEval in 12 languages.
- Surprisingly, StarCoder outperforms StarCoderBase on 9 languages in addition to Python.

Language	code-cushman-001	StarCoder	StarCoderBase
cpp	30.59	**31.55**	30.56
c-sharp	**22.06**	21.01	20.56
d	6.73	**13.57**	10.01
go	19.68	17.61	**21.47**
java	**31.90**	30.22	28.53
julia	1.54	**23.02**	21.09
javascript	31.27	30.79	**31.70**
lua	26.24	23.89	**26.61**
php	**28.94**	26.08	26.75
perl	**19.29**	17.34	16.32
python	30.71	**33.57**	30.35
r	10.99	**15.50**	10.18
ruby	**28.63**	1.24	17.25
racket	7.05	0.07	**11.77**
rust	**25.22**	21.84	24.46
scala	27.62	27.61	**28.79**
bash	**11.74**	10.46	11.02
swift	22.12	**22.74**	16.74
typescript	31.26	**32.29**	32.15

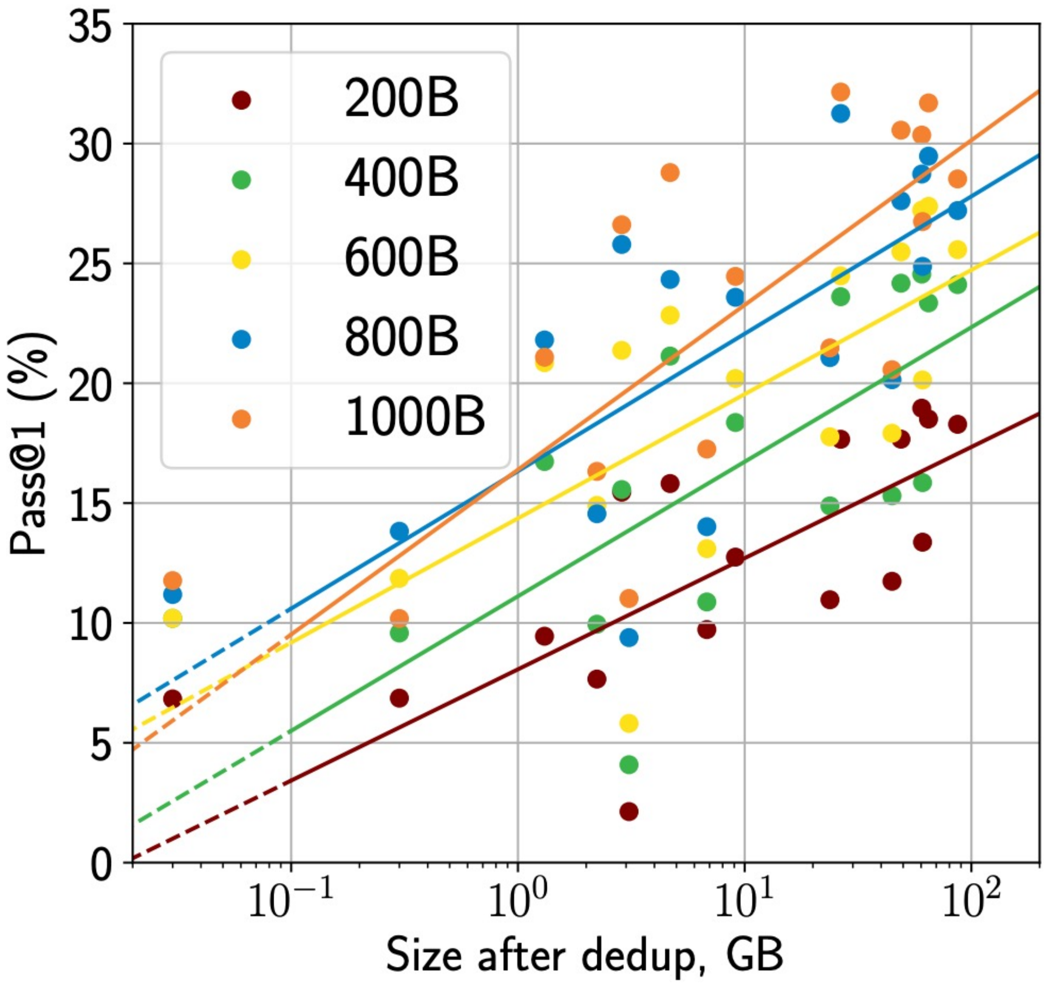
MultiPL-E translated HumanEval results

StarCoderBase: Performance Over Training



StarCoderBase: Performance By Data

- How correlated is code completion performance for a language with the amount of data available for a language?
- Train model for 200B tokens (on all languages). Evaluate on all languages, getting a dot for each language. Observe a strong correlation.
- Continue training, evaluate again at 400B tokens. The correlation remains strong, and line shifts upward.



DS-1000: Practical data tasks requiring API use

Problem

Here is a sample dataframe:

```
df = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6]})
```

I'd like to add inverses of each existing column to the dataframe and name them based on existing column names with a prefix, e.g. inv_A is an inverse of column A and so on.

The resulting dataframe should look like so:

```
result = pd.DataFrame({"A": [1, 2, 3], "B": [4, 5, 6], "inv_A": [1/1, 1/2, 1/3], "inv_B": [1/4, 1/5, 1/6]})
```

Obviously there are redundant methods like doing this in a loop, ***but** there should exist much more pythonic ways of doing it* … [omitted for brevity]

Code Context

```
A:
<code>
import pandas as pd
df = pd.DataFrame({"A": [1, 2, 3],"B": [4, 5, 6]})
</code>
BEGIN SOLUTION
<code>
[insert]
</code>
END SOLUTION
<code>
print(result)
</code>
```

Reference Solution

```
result = df.join(df.apply(lambda x: 1/x).add_prefix("inv_"))
```

Format	Model	Matplotlib	NumPy	Pandas	PyTorch	SciPy	Scikit-Learn	TensorFlow	Overall
	Number of problems:	155	220	291	68	106	115	45	1,000
Completion	InCoder-6B	28.3	4.4	3.1	4.4	2.8	2.8	3.8	7.4
Completion	CodeGen-16B-Mono	31.7	10.9	3.4	7.0	9.0	10.8	15.2	11.7
Completion	code-cushman-001	40.7	21.8	7.9	12.4	11.3	18.0	12.2	18.1
Completion	StarCoderBase	47.0	27.1	10.1	19.5	**21.7**	27.0	20.5	23.8
Completion	StarCoder	**51.7**	**29.7**	**11.4**	**21.4**	20.2	**29.5**	**24.5**	**26.0**

Evaluating Infilling

Model	Java	JavaScript	Python
InCoder-6B	0.49	0.51	0.31
SantaCoder	0.62	0.60	0.44
StarCoder	**0.73**	**0.74**	**0.62**

Single-line code completion for three languages (SantaCoder/InCoder benchmarks)

Model	BLEU
InCoder-6B	18.27
SantaCoder	19.74
StarCoderBase	21.38
StarCoder	**21.99**

Python docstring generation (CodeXGLUE / InCoder benchmark)

	Packages type check		
	✓	Total	%
InCoder	30	128	23.4
StarCoderBase	49	128	38.3

TypeScript type inference (TypeWeaver benchmarks)

Model	Non-None F1	All F1
InCoder-6B	59.1	46.8
SantaCoder	66.9	78.5
StarCoderBase	**77.4**	**86.6**
StarCoder	77.1	86.4

Python return-type prediction (InCoder/TypeWriter benchmarks)

Testing 8K Window: Perplexity with Long Contexts

Window Size	Language									
	cpp	c-sharp	c	go	java	javascript	php	r	ruby	rust
2K tokens	2.01	1.90	1.71	1.35	1.65	1.98	1.73	1.72	2.16	1.84
8K tokens	**1.79**	**1.66**	**1.61**	**1.21**	**1.54**	**1.68**	**1.43**	**1.48**	**2.02**	**1.65**

- Derived test data from GPL repositories on GitHub. GPL was excluded from training data.
- Demonstrates StarCoder can benefit from information within long files or repositories.
- Longer contexts provides noticeable decreases in perplexity.

Non-Trivial Natural Language Abilities

- Surprisingly reasonable performance on some natural language reasoning tasks
- CodeGen < StarCoderBase < LLaMA

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?
Solution: Beth bakes 4 2 dozen batches of cookies for a total of 4*2 = <<4*2=8>>8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of 12*8 = <<12*8=96>>96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat 96/16 = <<96/16=6>>6 cookies
Final Answer: 6

Model	Size	GSM8K CoT	+maj1@100	GSM8K PAL	+maj1@40
StarCoderBase	15.5B	8.4	—	21.5	31.2
CodeGen-Multi	16B	3.18	—	8.6	15.2
CodeGen-Mono	16B	2.6	—	13.1	22.4
LLaMA	7B	11.0	18.1	10.5	16.8
	13B	17.8	29.3	16.9	28.5
	33B	35.6	53.1	38.7	50.3
	65B	**50.9**	**69.7**	—	—

Reasoning Tasks in HELM

Model	Size	Open Access	Synth. Reason. (AS)	Synth. Reason. (NL)	bAbI	Dyck	GSM8K	MATH	MATH (CoT)	LSAT	Legal Support
code-davinci-002	175B		**54.0**	68.4	**68.6**	80.5	**56.8**	**41.0**	43.3	—	—
text-davinci-003	175B		50.2	**73.4**	65.3	75.1	50.6	39.0	**44.9**	**23.3**	**62.2**
Luminous Supreme	70B		31.2	—	50.4	72.9	11.2	14.9	5.7	21.2	53.0
StarCoderBase	15.5B	✓	44.0	21.0	50.4	**85.4**	8.4	15.1	7.0	19.0	53.2
Cohere Command Beta	52.4B		24.3	24.5	47.3	42.1	13.8	13.3	7.5	22.9	60.6
J1-Jumbo v1	178B		26.3	17.4	54.3	44.5	5.4	8.9	3.3	23.2	48.4
J1-Grande v2 beta	17B		28.6	13.9	47.0	61.7	9.6	12.7	6.8	19.1	56.2
code-cushman-001	12B		34.1	16.4	48.1	45.1	4.9	9.9	7.2	—	—
OPT	175B	✓	22.5	24.8	50.7	49.4	4.0	6.5	2.6	22.0	53.2
GPT-NeoX	20B	✓	20.4	16.7	46.8	74.7	5.3	14.1	7.1	19.1	51.5
BLOOM	176B	✓	30.4	19.7	44.7	54.5	9.5	4.3	5.5	20.9	54.3
GLM	130B	✓	25.2	25.4	44.3	54.9	6.1	0	5.9	19.3	45.1
UL2	20B	✓	20.5	21.7	50.1	14.0	2.4	0	0	20.7	50.6
OPT	66B	✓	19.3	21.3	40.8	47.1	1.8	4.8	2.9	17.5	52.7
YaLM	100B	✓	5.6	6.1	34.6	63.3	0	0	0	2.3	48.4
T5	11B	✓	19.6	10.1	41.2	34.7	2.3	0	0	15.9	55.8

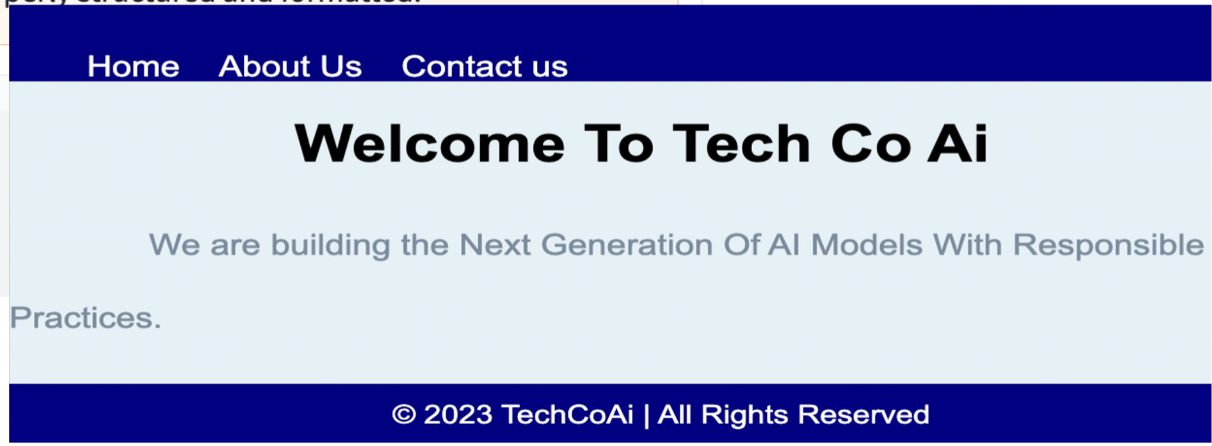
Chat Models via Continued Training

- StarChat-Beta: Continued training on Falcon English Web, the Pile, and OpenAssistant Dataset

Chat

Create a simple HTML website for a fictional company called "TechCoAI". Please make an HTML file that adheres to the following specifications:
The webpage should have a title that says "TechCoAI - Welcome".
The background color of the webpage should be light blue (#E6F1F6).
The website should have a navigation menu with three links: "Home" "About" and "Contact ", these links should be displayed in dark blue.
Include a paragraph of text that describes TechCoAI's mission and vision: " The mission of TechCoAI is to build the next generation of AI models with responsible practices."
Finally, create a footer with the copyright information.
The footer should display the copyright symbol (©), the current year 2023, and "TechCoAI. All rights reserved."
Remember to follow these instructions precisely and ensure that the HTML code is properly structured and formatted.

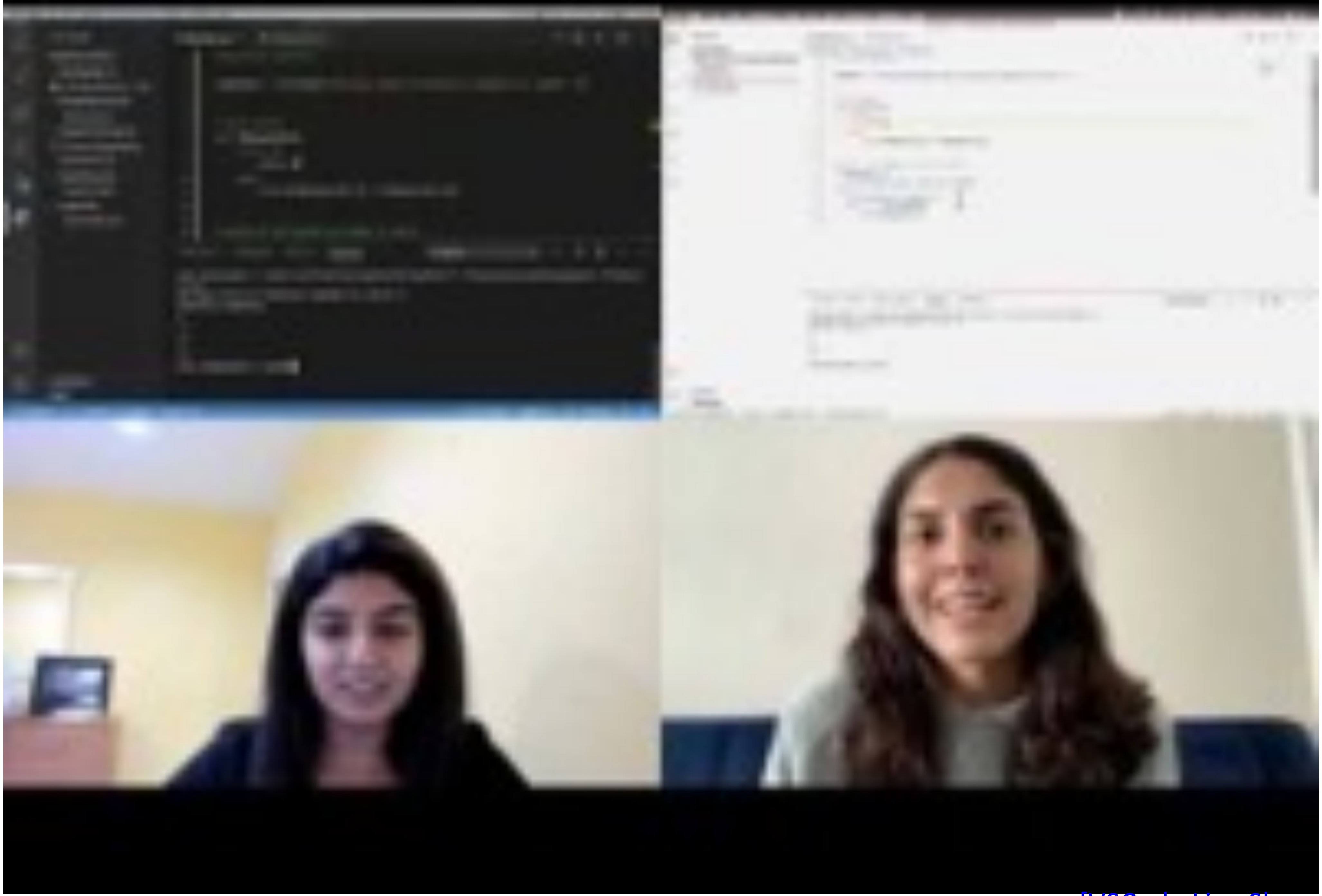
```
<html lang="en">
  <head>
    <!-- Required meta tags -->
      <meta charset="utf-8">
```



https://huggingface.co/spaces/HuggingFaceH4/starchat-playground

Programming as Communication

We begin with the conjecture that most software… is created by humans at work,
~~with all the attendant constraints and limitations~~
communicating with the compiler, other developers, and themselves,
and thus, like natural language,
~~it is also likely to be repetitive and predictable.~~
writing software is a form of contextual and interactive communication.
We then proceed to ask whether a) code can be usefully modeled by statistical language models and b) such models can be leveraged to support software engineers.



[VSCode Live Share Demo, 2020]

Communicating with Multiple Modalities

	Natural Language	Partial Code	Tests & Execution	Edits to Code	Deictic (Pointing / Highlighting)	…
As Inputs	InCoder, SantaCoder, Starcoder	InCoder, SantaCoder, Starcoder	MBR-Exec [Shi et al. 2022]			
As Outputs	InCoder, SantaCoder, Starcoder			[Wallace et al., in progress]		

Modality Choice
[Lin et al., 2022]

Communicating with Multiple Modalities

	Natural Language	Partial Code	Tests & Execution	Edits to Code	Deictic (Pointing / Highlighting)	…
As Inputs	InCoder, SantaCoder, Starcoder	InCoder, SantaCoder, Starcoder	MBR-Exec [Shi et al. 2022]			
As Outputs	InCoder, SantaCoder, Starcoder			[Wallace et al., in progress]		

Modality Choice
[Lin et al., 2022]

Using Test Inputs

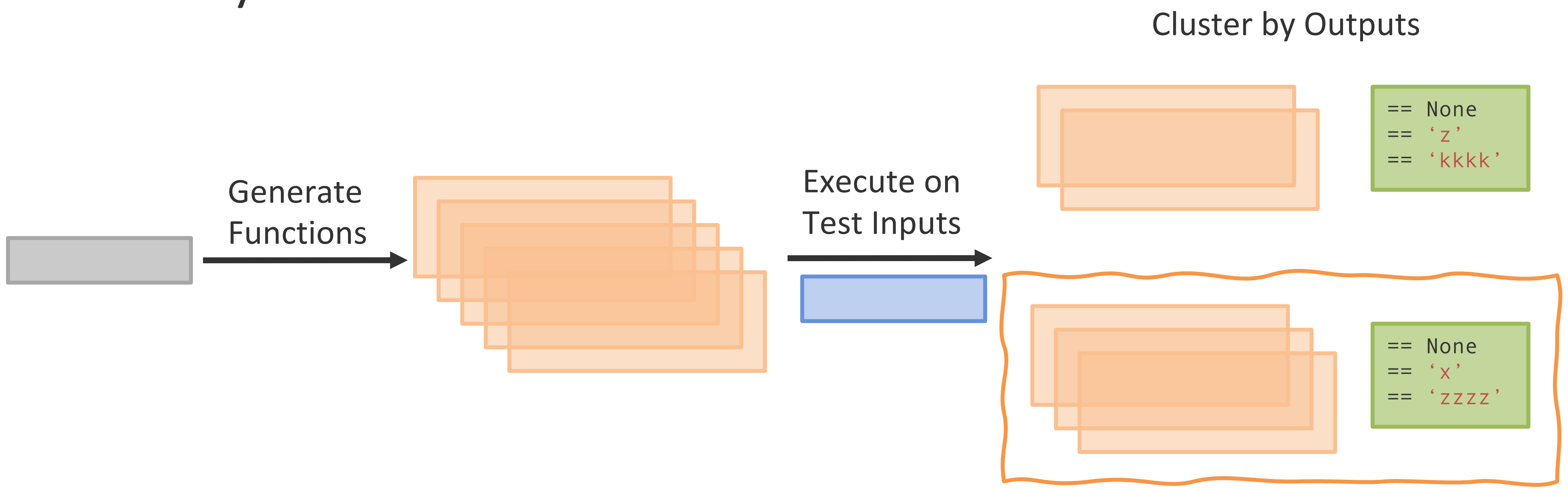
Description:

```python
def longest(strings: List[str]) -> Optional[str]:
  """ Out of list of strings, return the longest one.
  Return the first one in case of multiple strings of
  the same length. Return None if the list is empty."""
```

Test Inputs:

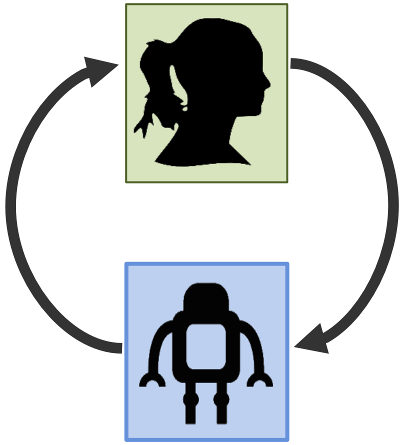
```python
longest([]) == ___
longest(['x', 'y', 'z']) == ___
longest(['x', 'yyy', 'zzzz', 'www', 'kkkk', 'abc']) == ___
```

Minimum Bayes Risk with Execution:



[Shi et al. 2022. See also *AlphaCode*, Li et al. 2022]

Other Features of Communication

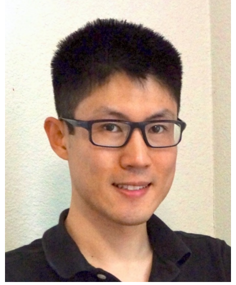


- Communicative cost
  - Copilot outputs can be hard to understand [Vaithilingam et al. 2022]
  - Would a user rather type a comment or edit code?
- Resolving uncertainty
  - Disambiguate by prompting with test inputs [Zhong et al. 2022]
  - How to convey uncertainty to the user & build trust?
- Adaptation
  - Acceleration vs exploration modes for using Copilot [Barke et al. 2022]
  - API preferences, functional vs imperative, design patterns, documentation style …

Collaborators

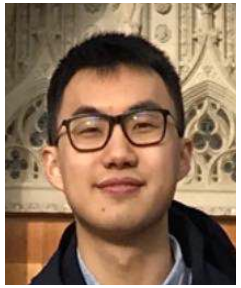

Armen Aghajanyan


Jessy Lin


Sida Wang


Eric Wallace


Freda Shi


Ruiqi Zhong


Scott Yih


Luke Zettlemoyer


Mike Lewis


Raymond Li


Louba Ben Allal


Denis Kocetkov


Arjun Guha


Leandro von Werra


Harm de Vries

and ~60 others from the BigCode project!

BigCode

Meta