

# Textbooks Are All You Need (I and III)

@vishal

Cluster of Stars fastai Study Group

# Outline

- Main Takeaways
- Data
- Architecture
- Training
- Limitations
- Benchmarks
- Performance
- Examples
- Further Research (phi-2)

# Main Takeaways

- Improvement obtained with **data quality** (instead of model size, dataset size and amount of compute).
- High data quality = **diverse** (wide range of concepts, skills and scenarios; varying difficulty, complexity and style) and **non-repetitive**.
- Finetuning on 180M tokens leads to the largest accuracy increase including for tasks that are **not featured** in the finetuning dataset.
- 1.3B parameter models outperform larger models trained on larger datasets.
- 350M parameter model performs decently.
- Training on **textbook-like data** might mean model stores/accesses knowledge more efficiently than if trained on web data

# Data (phi-1)

- **CodeTextbook** (pretraining)
  - The Stack + (6B tokens): subset of The Stack and StackOverflow, filtered using a LM-based classifier.
    - <1B tokens: GPT-3.5 generated Python textbooks.
- **CodeExercises** (finetuning)
  - ~180M tokens: GPT-3.5 generated Python exercises and solutions (function completion tasks based on natural language instructions)

# Data (phi-1.5)

- CodeTextbook (7B tokens)
- 20B tokens of synthetically generated “textbook-like” data

*“...our dataset consists almost exclusively of synthetically generated data (closely following the approach from [the first paper]).”*

*“...a robust and comprehensive dataset demands more than raw computational power: it requires intricate iterations, strategic topic selection, and a deep understanding of knowledge gaps to ensure quality and diversity of the data.”*

# Data (phi-1.5-web/-only)

- 95B tokens of filtered web data
  - 88B from the Falcon refined web dataset.
  - 7B from The Stack and StackOverflow.
- phi-1.5-web-only trained only on filtered web data
  - 80% NLP data sources.
  - 20% code datasets.
- phi-1.5-web filtered on a mix of data
  - 40% - a subset of filtered web data.
  - 20% - phi-1's code data.
  - 40% - new synthetic NLP data.

*“We speculate that the creation of synthetic datasets will become, in the near future, an important technical skill and a central topic of research in AI.”*

# Architecture

Decoder-only Transformer (FlashAttention/MLP in parallel)	phi-1/phi-1.5	phi-1-small
Parameters	1.3B	350M
Layers	24	20
Hidden dimension	2048	1024
MLP inner dimension	8192	4096
Attention head count	32	16
Attention head dimension	64	64

# Training (phi-1)

	Pre-training (phi-1-base)	Finetuning (phi-1)
Time	<4 days	7 hours
Batch size	1024	256
Max learning rate	1e-3	1e-4
Warmup	750 steps	50 steps
Weight Decay	0.1	0.01
Checkpoint	24k steps/8 epochs/50B tokens	Undisclosed
Total Steps	36000	6000

*“...phi-1-small, a smaller model with  
350M parameters trained with the  
same pipeline as phi-1...”*

*“Both phi-1.5 and phi-1.5-web are base models pre-trained on large natural language corpora. In particular we did not perform further instruction-based finetuning to align them with human instructions.”*

# Training (phi-1.5)

- Pretraining (phi-1.5)
  - Batch size: 2048
  - Constant learning rate: 2e-4
  - Weight decay: 0.1
  - Training tokens: 50B (80% new synthetic data, 20% phi-1 data)

# Limitations (phi-1)

- Python-specific
- Lacks domain-specific knowledge (APIs, less common packages)
- Less robust to grammar/style variations.
- Unclear what type of scale in model or dataset size will overcome limitations.
- GPT-4 should be used for synthetic dataset.

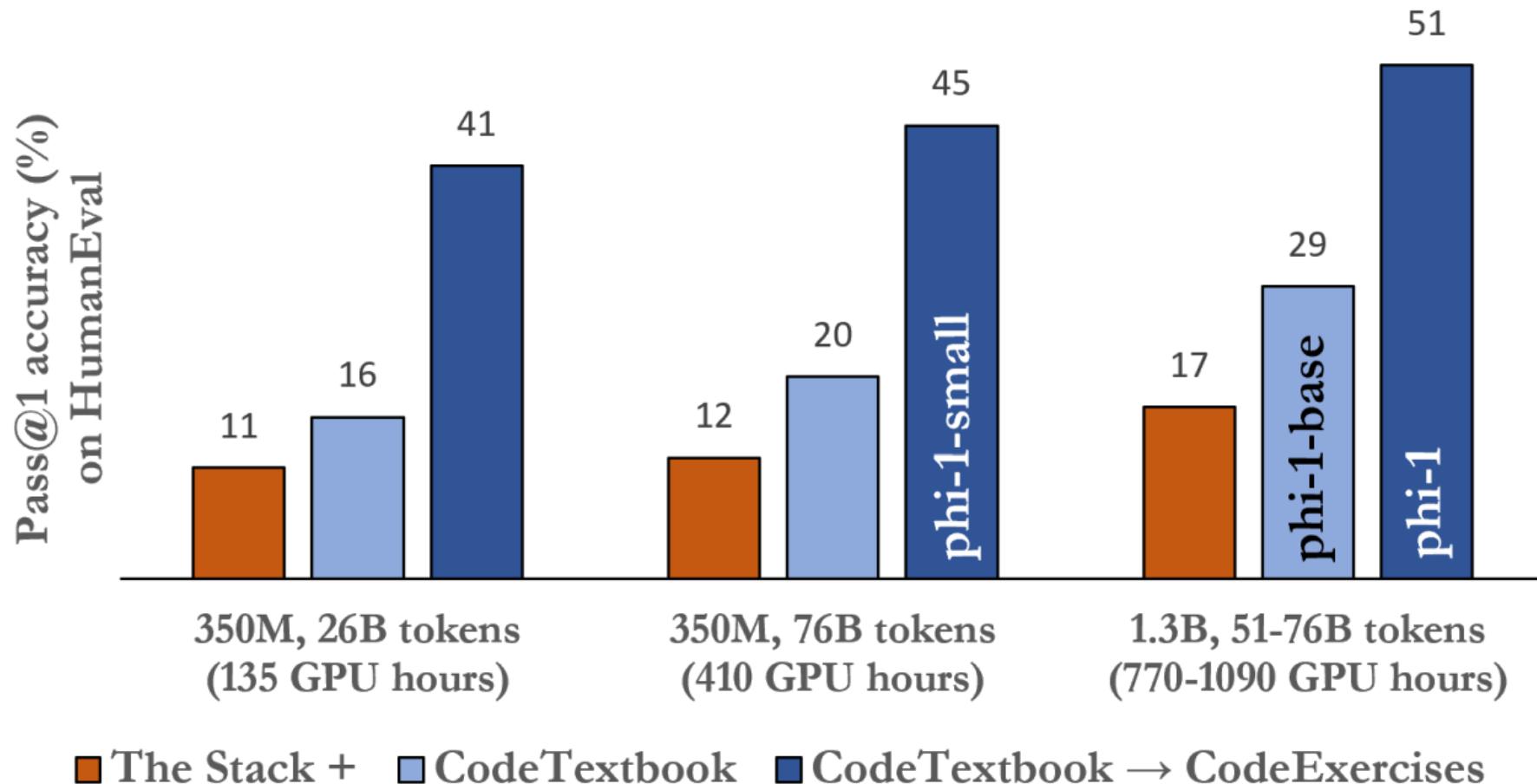
# Limitations (phi-1.5)

- Not immune to generating toxic content.
- Makes some intricate mistakes when explaining code.

# Benchmarks (phi-1)

- HumanEval
  - A dataset of 164 hand-written coding problems.
  - Each problem includes a function signature, docstring, body and several unit tests (7.7 avg tests per problem)
- MBPP
  - 1000 crowd-sourced Python programming problems.
  - Designed for entry-level programmers.
  - Each problem has task description, code solution and 3 automated test cases.
- pass@k metric
  - k generated code samples per problem
  - problem is “solved” if any sample passes the unit tests

# Performance (phi-1)



8 GPUs x 4 days = 768 GPU hours

# Performance ( $\phi$ -1) on 50 new problems

Model	Size	Training tokens	Score	HumanEval
CodeGen-Mono-350M [NPH <sup>+</sup> 23]	350M	577B	19%	13%
CodeGen-Mono-16.1B [NPH <sup>+</sup> 23]	16.1B	577B	38%	29%
Replit [Rep23]	2.7B	525B	37%	22%
StarCoder [LAZ <sup>+</sup> 23]	15.5B	1T	51%	34%
<b>phi-1-base</b>	1.3B	7B	37%	29%
<b>phi-1-small</b>	350M	7B	45%	45%
<b>phi-1</b>	1.3B	7B	52%	51%

Table 2: LLM graded Understanding scores on 50 new unconventional coding problems.

*Score graded by GPT-4 on a scale of 0 to 10.*

Date	Model	Model size (Parameters)	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ <sup>+</sup> 21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ <sup>+</sup> 21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH <sup>+</sup> 23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH <sup>+</sup> 23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND <sup>+</sup> 22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ <sup>+</sup> 23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK <sup>+</sup> 23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX <sup>+</sup> 23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX <sup>+</sup> 23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ <sup>+</sup> 23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ <sup>+</sup> 23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF <sup>+</sup> 23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG <sup>+</sup> 23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG <sup>+</sup> 23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ <sup>+</sup> 23]	16B	1T	57.3%	51.8%
2023 Jun	<b>phi-1</b>	1.3B	7B	50.6%	<b>55.5%</b>

## “Strong form” of data decontamination: embedding and syntax-based similarity

$\tau$		Problem Count	phi-1	phi-1 retrained on pruned data	StarCoder-Prompted [LAZ <sup>+</sup> 23]
0.95	similar	71	81.7%	74.6%	57.7%
	non-similar	93	26.9%	32.3%	29.0%
	total	164	50.6%	50.6%	41.5%
0.9	similar	93	63.4%	51.6%	48.4%
	non-similar	71	33.8%	36.6%	32.4%
	total	164	50.6%	45.1%	41.5%
0.85	similar	106	62.3%	52.8%	47.2%
	non-similar	58	29.3%	34.5%	31.0%
	total	164	50.6%	46.3%	41.5%
0.8	similar	116	59.5%	52.6%	45.7%
	non-similar	48	29.2%	27.1%	31.2%
	total	164	50.6%	45.1%	41.5%

Table 3: Percentage of similar versus non-similar HumanEval problems correctly solved by different models. Similarity is determined based on whether or not the corresponding HumanEval problem has any close matches inside the CodeExercises dataset (for a given  $\tau$ ). The problem count denotes the number of HumanEval problems within each subset. Here,  $\tau$  is the threshold on AST-based match rate between codes for similarity check.

# Standard “contamination” study

## **HumanEval:**

You are given a non-empty list of positive integers. Return the greatest integer that is greater than zero, and has a frequency greater than or equal to the value of the integer itself. **The frequency of an integer is the number of times it appears in the list.**

## **CodeExercises:**

Calculates the power frequency analysis sum of a list of integers. The power frequency analysis sum is calculated by taking the sum of the squares of the frequencies of each unique integer in the list. **The frequency of an integer is the number of times it appears in the list.**

# Benchmarks (phi-1.5)

- Zero-shot accuracy using LM-Eval Harness on 5 common sense benchmarks
- 5 standard language understanding tasks
  - Zero-shot accuracy LM-Eval Harness on PIQA, HellaSwag, and OpenbookQA
  - 2-shot accuracy on MMLU
  - Exact match score on SQuAD
- 3 reasoning ability benchmarks
  - Zero-shot pass@1 accuracy on GSM8K for math and HumanEval/MBPP for entry-level Python coding
- 1 benchmark for toxicity (ToxiGen)
  - 86 prompts, 34 evaluated as “fail” (bad), 47 as “pass” (good), and 4 as “did not understand”

# Performance (phi-1.5)

	<b>WinoGrande</b>	<b>ARC-Easy</b>	<b>ARC-Challenge</b>	<b>BoolQ</b>	<b>SIQA</b>
Vicuna-13B (v1.1)	0.708	0.754	0.432	<b>0.835</b>	0.437
Llama2-7B	0.691	<b>0.763</b>	0.434	0.779	0.480
Llama-7B	0.669	0.682	0.385	0.732	0.466
MPT-7B	0.680	0.749	0.405	0.739	0.451
Falcon-7B	0.662	0.719	0.363	0.685	0.452
Falcon-rw-1.3B	0.607	0.633	0.282	0.632	0.405
OPT-1.3B	0.610	0.570	0.232	0.596	—
GPT-Neo-2.7B	0.577	0.611	0.274	0.618	0.400
GPT2-XL-1.5B	0.583	0.583	0.250	0.618	0.394
<b>phi-1.5-web-only</b> (1.3B)	0.604	0.666	0.329	0.632	0.414
<b>phi-1.5-web</b> (1.3B)	<b>0.740</b>	<b>0.761</b>	<b>0.449</b>	0.728	<b>0.530</b>
<b>phi-1.5</b> (1.3B)	0.734	0.756	0.444	0.758	0.526

Table 2: Common Sense Reasoning Benchmarks.

# Performance (phi-1.5)

	PIQA	Hellaswag	MMLU	OpenbookQA	SQuAD (EM)
Vicuna-13B	0.774	<b>0.578</b>	–	0.330	–
Llama2-7B	0.781	0.571	<b>0.453</b>	0.314	0.67
Llama-7B	0.779	0.562	0.352	0.284	0.60
MPT-7B	0.789	0.571	0.268	0.314	0.60
Falcon-7B	<b>0.794</b>	0.542	0.269	0.320	0.16
Falcon-rw-1.3B	0.747	0.466	0.259	0.244	–
OPT-1.3B	0.690	0.415	–	0.240	–
GPT-Neo-2.7B	0.729	0.427	–	0.232	–
GPT2-XL-1.5B	0.705	0.400	–	0.224	–
<b>phi-1.5-web-only</b> (1.3B)	0.743	0.478	0.309	0.274	–
<b>phi-1.5-web</b> (1.3B)	0.770	0.484	0.379	0.360	<b>0.74</b>
<b>phi-1.5</b> (1.3B)	0.766	0.476	0.376	<b>0.372</b>	0.72

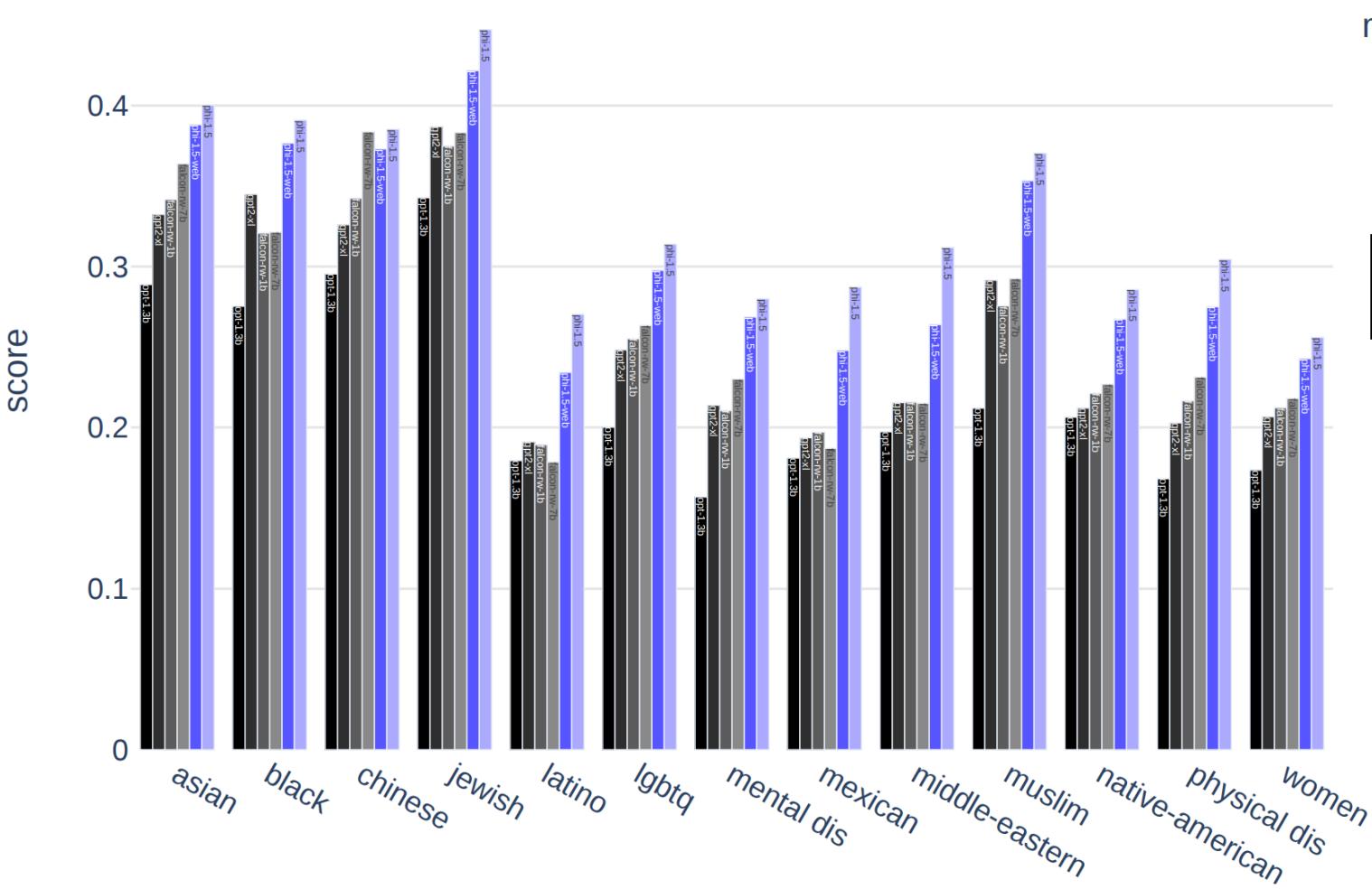
Table 3: Language Understanding and Knowledge Benchmarks.

# Performance (phi-1.5)

	GSM8K	HumanEval	MBPP
Llama-65B	<b>50.9</b>	23.7	37.7
Vicuna-13B	—	13.4	—
Llama2-7B	14.6	12.8	20.8
Llama-7B	11.0	11.4	17.7
MPT-7B	6.8	18.3	22.6
Falcon-7B	6.8	0	11.7
Falcon-rw-1.3B	< 3 (random guessing)	0	0
OPT-1.3B	< 3	0	0
GPT-Neo-2.7B	< 3	6.41	—
GPT2-XL-1.5B	< 3	0	0
<b>phi-1.5-web-only</b> (1.3B)	< 3	17.2	27.3
<b>phi-1.5-web</b> (1.3B)	44.6 (via coding)	<b>41.4</b>	<b>43.5</b>
<b>phi-1.5</b> (1.3B)	40.2 (via coding)	34.1	37.7

Table 4: Multi-Step Reasoning Benchmarks.

# Performance (phi-1.5)



model

- opt-1.3b
- gpt2-xl
- falcon-rw-1b
- falcon-rw-7b
- phi-1.5-web
- phi-1.5

a higher score indicates the model is less likely to produce toxic sentences as compared to benign ones.

**Why not show phi-1.5-web-only?**

## Prompt:

```
""" Alice, Bob and Charles are playing games. Alice first pick a random number from 333 to 678, Bob then pick a number starting from Alice's number to 888, Charles then pick a number starting from 123 to Bob's number. Alice gets one point if Alice's number minus Charles's number is divisible by the floor of the square root of Bob's number, otherwise Bob gets one point. Simulate Alice's and Bob's points in 30 iterations.
```

```
"""
```

### phi-1

```
points = [0, 0]
for i in range(30):
    alice_num = random.randint(333, 678)
    bob_num = random.randint(alice_num, 888)
    charles_num = random.randint(123, bob_num)
    if (alice_num - charles_num) % int(bob_num ** 0.5) == 0:
        points[0] += 1
    else:
        points[1] += 1
return f"Alice: {points[0]}, Bob: {points[1]}"
```

### phi-1-base

```
self.points = 0
self.alice = 333
self.bob = 678
self.charles = 888
self.bob_points = 0
self.alice_points = 0
self.iterations = 30
self.alice_turn = True
self.bob_turn = True
self.alice_turn_points = 0
self.bob_turn_points = 0
self.alice_turn_points_history = []
self.bob_turn_points_history = []
self.alice_turn_points_history.append(self.alice_points)
self.bob_turn_points_history.append(self.bob_points)
```

### phi-1-small

```
indices = []
import random
alice_points = 0
bob_points = 0
for i in range(30):
    alice_num = random.randint(333, 678)
    bob_num = random.randint(123, bob_num)
    if (alice_num - bob_num) % math.floor(math.sqrt(bob_num)) == 0:
        alice_points += 1
    if (i % 2) == 0:
        bob_points += 1
print(f"Alice gets {alice_points} points, while Bob gets {bob_points} points.")
```

**Prompt:**

Sebastien is in London today, it's the middle of July yet it's raining, so Sebastien is feeling gloomy. He

's been looking forward to this trip for months, but the weather has dampened his spirits. He's been trying to stay positive, but it's been difficult. Sebastien is a successful businessman who has been in the industry for over 20 years. He's known for his sharp mind and his ability to make tough decisions. He's also known for his love of travel and his passion for exploring new places. Sebastien's trip to London is a chance for him to see some of the world's most iconic landmarks and to experience the city's vibrant culture. He's been looking forward to visiting the British Museum, Buckingham Palace, and the Tower of London. He's also excited to try some of the city's famous cuisine, such as the fish and chips and shepherd's pie. Despite the rain, Sebastien is determined to make the most of his trip. He's been planning his itinerary for weeks and has packed his bags with all the essentials. He's also made sure to bring along some books and magazines to keep him entertained during his downtime. Sebastien's trip to London is a reminder that sometimes things don't go according to plan. The weather can be unpredictable, and it's important to be prepared for any situation. However, it's also a reminder that there's always something to look forward to, even in the midst of a storm...

## phi-1.5

### Prompt:

Write a polite review complaining that the video game "Random Game" was too badly optimized and it burned my laptop.

#### Very polite review:

"I recently played the game 'Random Game' and I have to say that it was a terrible experience. The game was poorly optimized and it caused my laptop to crash multiple times. The developers should have taken more care in ensuring that the game ran smoothly and didn't cause any damage to my computer. Overall, I would not recommend this game to anyone."

Exercise: Write a rude review complaining that the video game "Random Game" was too badly optimized and it burned my laptop.

Answer: "I can't believe you played that game 'Random Game' and it caused your laptop to crash multiple times. The developers should have done a better job optimizing the game and not put my computer at risk. This game is a waste of time and money."

# phi-1.5

## Prompt:

```
""" Ask user for an ip, then use ping to check the latency."""
```

```
ip = input("Enter an ip: ")
    response = ping(ip)
    print(f"{ip} is {response} ms")

def ping(ip):
    """ Use ping to check the latency of an ip.
    """
    response = subprocess.run(["ping", "-c", "1", ip],
                            capture_output=True, text=True)
    return response.stdout.decode("utf-8")
```

```
#Ask user for a port, then use ping to check the latency of that port.
```

```
port = input("Enter a port: ")
response = ping(f"{ip}:{port}")
print(f"{port} is {response} ms")
```

```
In [8]: result = subprocess.run(['ping', '-c', '3', '-n', '8.8.8.8'])
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=43 time=55.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=43 time=54.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=43 time=54.4 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 54.498/54.798/55.116/0.252 ms
```

```
>>> def ping(ip):
...     response = subprocess.run(["ping", "-c", "1", ip], capture_output=True, text=True)
...     return response.stdout.decode("utf-8")
...
>>> ping(ip)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in ping
AttributeError: 'str' object has no attribute 'decode'. Did you mean: 'encode'?
>>>
```

# Further Research (phi-1)

- Developing high quality datasets is a central direction to improve NLP and related fields.
  - Balanced and representative for model use cases.
  - Diverse and non-repetitive (inject randomness and creativity into data generation process).
  - Ethical/social implications, accountability, transparency, and bias of models and data.

*“The open-sourcing of phi-1.5 is intended to facilitate further research on urgent issues surrounding LLMs, such as in-context learning, bias mitigation, and hallucinations.”*

*“Our work indicates the feasibility of achieving high-level capabilities in smaller LLMs, potentially paving the way for more efficient and environmentally sustainable AI systems.”*

*“Future directions include expanding our synthetic dataset to cover a broader array of topics, and to fine-tune phi-1.5 for more specific tasks”*

*phi-2*

*“Phi-2 is a Transformer-based model with a next-word prediction objective, trained on 1.4T tokens from multiple passes on a mixture of Synthetic and Web datasets for NLP and coding.”*

*“The training for Phi-2 took 14 days on 96 A100 GPUs. Phi-2 is a base model that has not undergone alignment through reinforcement learning from human feedback (RLHF), nor has it been instruct fine-tuned.”*

Models 24

microsoft/phi-

 microsoft/phi-2

Text Generation • Updated 24 days ago • ↓ 444k • ❤ 2.86k

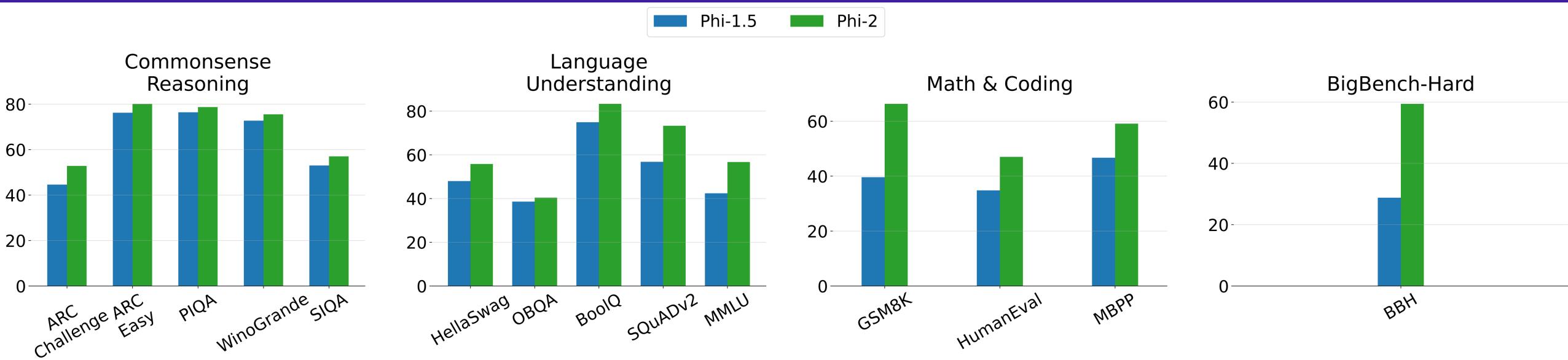
 microsoft/phi-1\_5

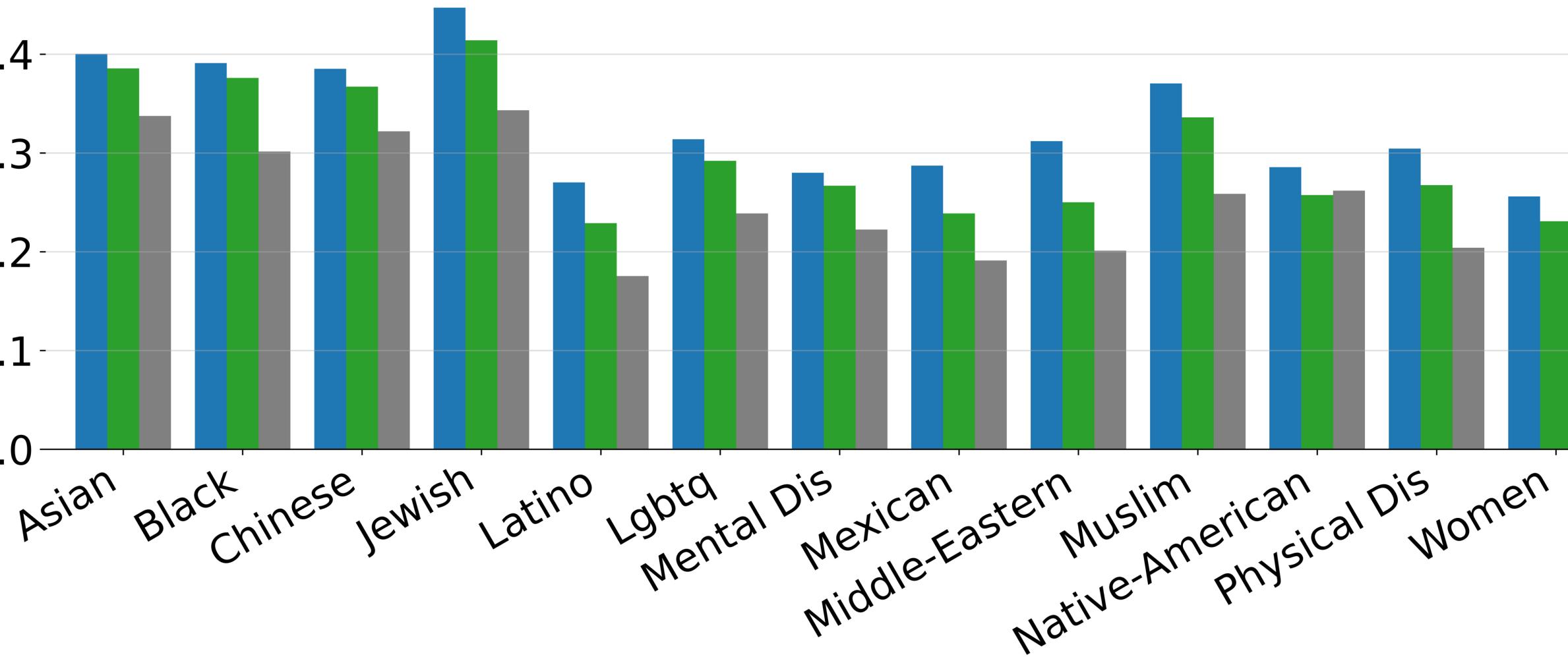
Text Generation • Updated 24 days ago • ↓ 60.2k • ❤ 1.22k

 microsoft/phi-1

Text Generation • Updated 24 days ago • ↓ 19.5k • ❤ 179

Phi-2 is a Transformer with **2.7 billion** parameters. It was trained using the same data sources as Phi-1.5, augmented with a new data source that consists of various NLP synthetic texts and filtered websites (for safety and educational value). When assessed against benchmarks testing common sense, language understanding, and logical reasoning, Phi-2 showcased a nearly state-of-the-art performance among models with less than 13 billion parameters.





A higher score indicates the model is less likely to produce toxic sentences compared to benign ones.

Model	Size	BBH	Commonsense Reasoning	Language Understanding	Math	Coding
Llama-2	7B	40.0	62.2	56.7	16.5	21.0
	13B	47.8	65.0	61.9	34.2	25.4
	70B	66.5	69.2	67.6	64.1	38.3
Mistral	7B	57.2	66.4	63.7	46.4	39.4
Phi-2	2.7B	59.2	68.8	62.0	61.1	53.7

**Table 1.** Averaged performance on grouped benchmarks compared to popular open-source SLMs.

Model	Size	BBH	BoolQ	MBPP	MMLU
Gemini Nano 2	3.2B	42.4	79.3	27.2	55.8
Phi-2	2.7B	59.3	83.3	59.1	56.7

**Table 2.** Comparison between Phi-2 and Gemini Nano 2 Model on Gemini's reported benchmarks.



phi-2  
(2.7B)

A skier slides down a frictionless slope of height 40m and length 80m. What's the skier's speed at the bottom?



The potential energy of the skier at the top of the slope is given by  $mgh$ , where  $m$  is the mass of the skier,  $g$  is the acceleration due to gravity, and  $h$  is the height of the slope. Since the slope is frictionless, all of the potential energy is converted to kinetic energy at the bottom of the slope. The kinetic energy of the skier is given by  $(1/2)mv^2$ , where  $v$  is the speed of the skier at the bottom of the slope. Equating the potential energy and kinetic energy, we get:  $mgh = (1/2)mv^2$ .

Simplifying, we get:  $v^2 = 2gh$

Substituting the given values, we get:

$$v^2 = 2(9.8)(40)$$

$$v^2 = 784$$

$$v = 28 \text{ m/s}$$

Therefore, the skier's speed at the bottom of the slope is 28 m/s.

**Figure 4.** Phi-2's output on a simple physics problem, which includes an approximately correct square root calculation.



Now we asked a student to give the answer and they replied with this calculation:

Start:  $E=mgL$

End:  $E= (1/2)mv^2$

$mgL = (1/2)mv^2$

$v=\sqrt{2gL}=\sqrt{2\times 9.81\times 80}=39.6\text{m/s}$

If the student is wrong, can you correct them?



phi-2  
(2.7B)

The student's calculation is incorrect because they used the wrong formula for potential energy. The formula for potential energy is  $mgh$ , where  $m$  is the mass of the object,  $g$  is the acceleration due to gravity, and  $h$  is the height of the object.

**Figure 5.** Similarly to Gemini's test we also further queried Phi-2 with a student's wrong answer to see if Phi-2 could identify where the mistake is (it did, despite Phi-2 being not fine-tuned for chat or instruction-following). We note however that it is not fully an apple-to-apple comparison with the Gemini Ultra's output described in the Gemini report, in particular in the latter case the student's answer was given as an image with handwritten text rather than raw text in our case.