OctoPack: Instruction Tuning Code Large Language Models

1) CommitPack

Code Before

```
import numpy as np
import matplotlib.pyplot as plt

# generate sample data
x_data = np.linspace(-5, 5, 20)
y_data = np.random.normal(0.0, 1.0, x_data.size)

plt.plot(x_data, y_data, 'o')
plt.show()
```

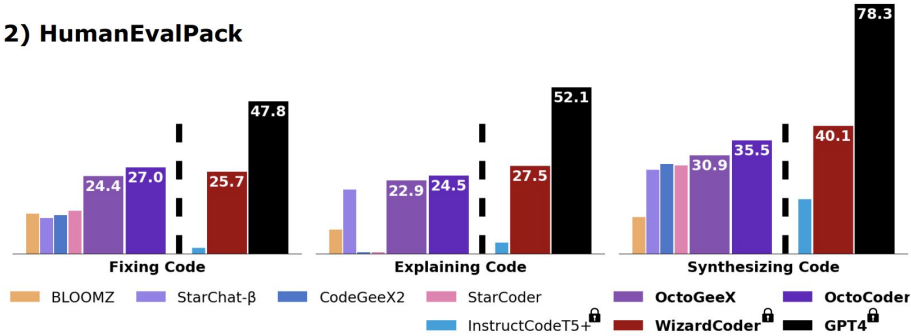
Commit Message

Change to sin() function with noise

Code After

```
import math
import numpy as np
import matplotlib.pyplot as plt

# generate sample data
x_data = np.linspace(-math.pi, math.pi, 30)
y_data = np.sin(x_data) + np.random.normal(0.0, 0.1, x_data.size)

plt.plot(x_data, y_data, 'o')
plt.show()
```

2) HumanEvalPack



Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, Shayne Longpre

ICLR

Motivation

Code LLMs only excel at code continuation

User

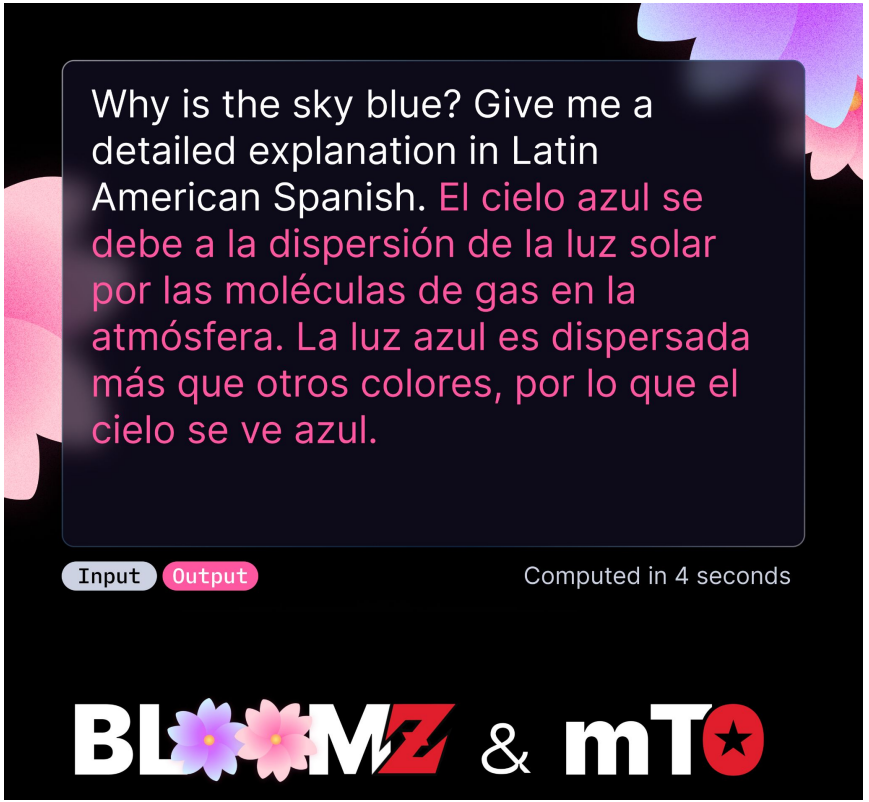
def sum(x, y): return x+y

Could you explain the function to me?

def sum(x, y): return x+y
def sum(x, y): return x+y

Code LLM

Success of instruction tuning non-code LLMs



Commits

Code Before

```
import numpy as np
import matplotlib.pyplot as plt

# generate sample data
x_data = np.linspace(-5, 5, 20)
y_data = np.random.normal(0.0, 1.0, x_data.size)

plt.plot(x_data, y_data, 'o')
plt.show()
```

Commit Message

Change to sin() function with noise

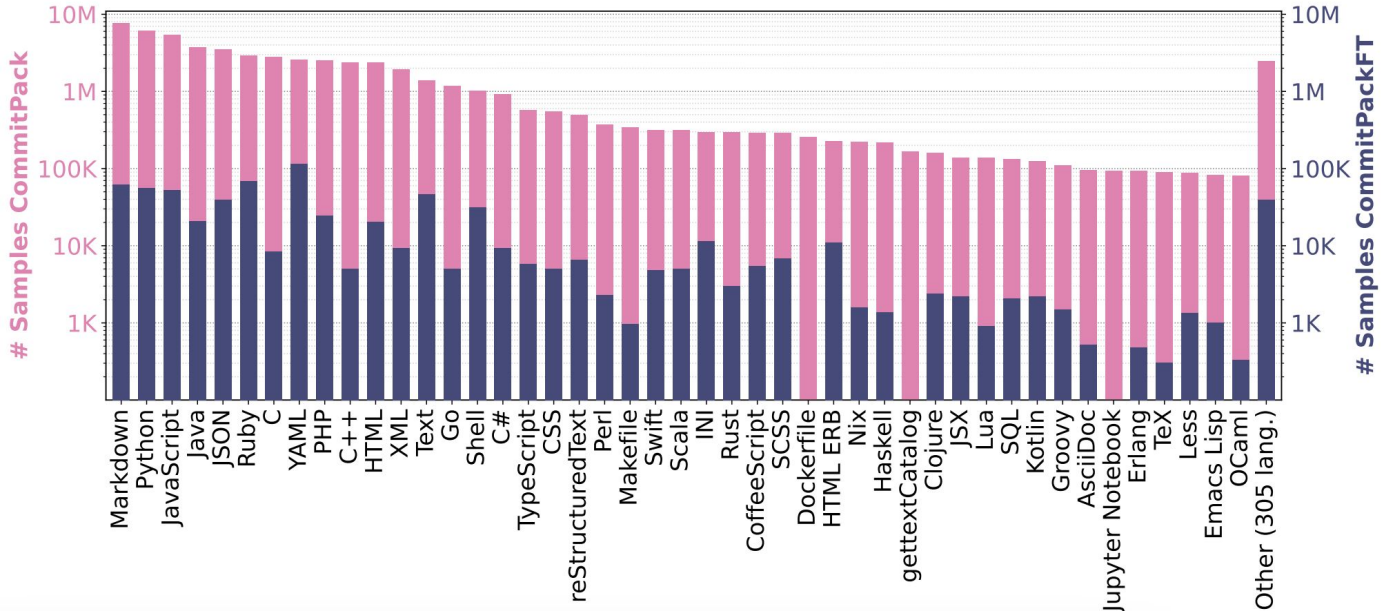
Code After

```
import math
import numpy as np
import matplotlib.pyplot as plt

# generate sample data
x_data = np.linspace(-math.pi, math.pi, 30)
y_data = np.sin(x_data) + np.random.normal(0.0, 0.1, x_data.size)

plt.plot(x_data, y_data, 'o')
plt.show()
```

CommitPack: Code Instruction Data

CommitPack - 4TB of scraped commits:

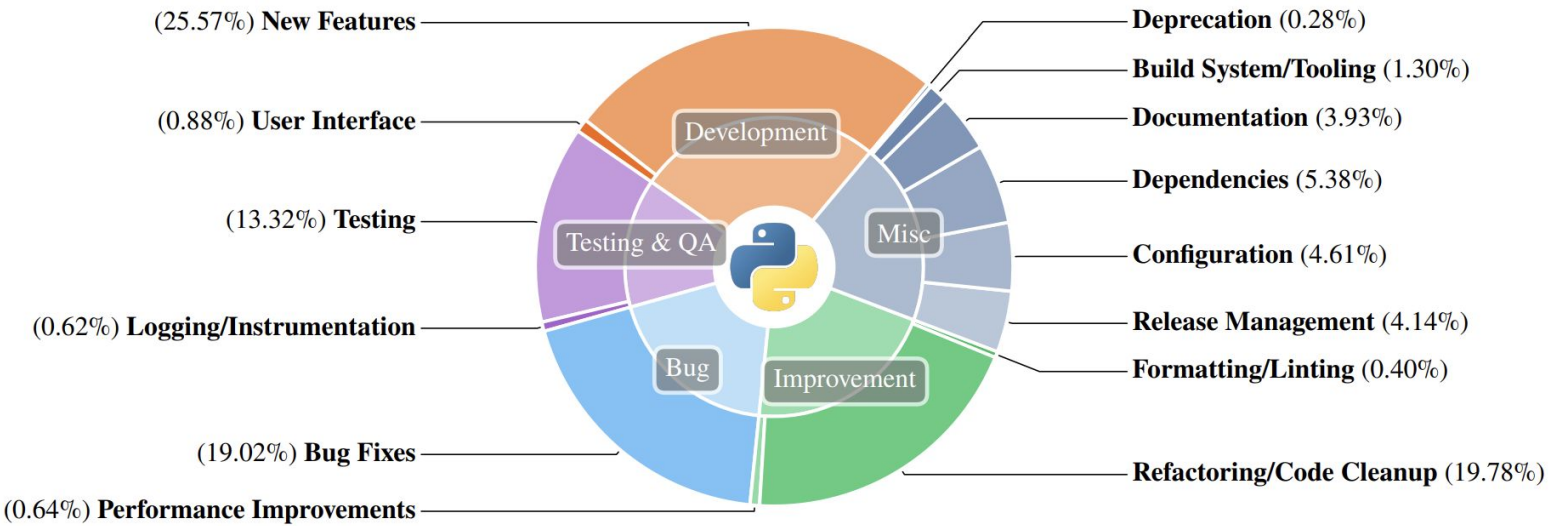
- Only commercially licensed code
- Only single-file commits to avoid complexity
- Filtered out low-quality commits (e.g. merges)



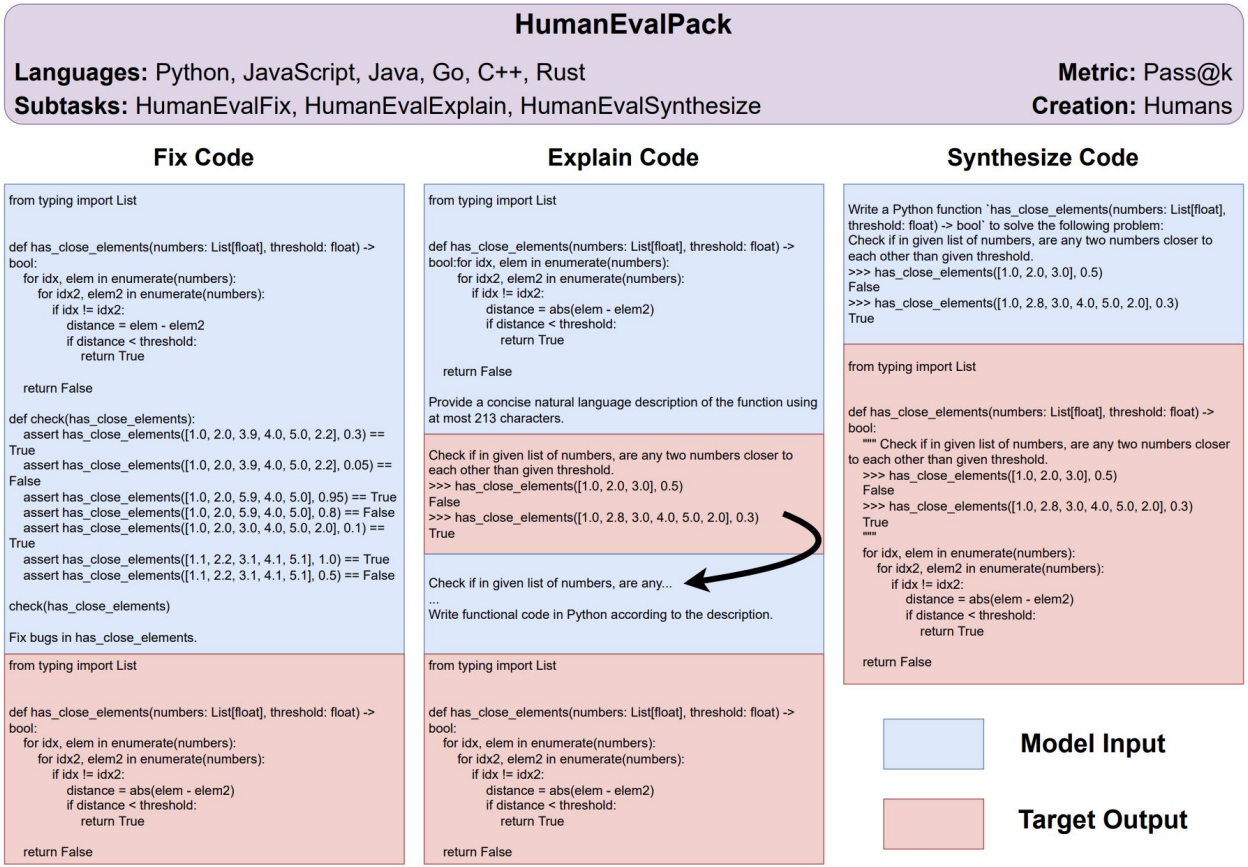
CommitPackFT: Code Instruction Data

CommitPackFT - 2GB high-quality selection:

- Only commits that resemble instructions (e.g. `Increase beta2 value in optimizer`)
- No commits with external references like issues
- Only detailed commits within certain word limits

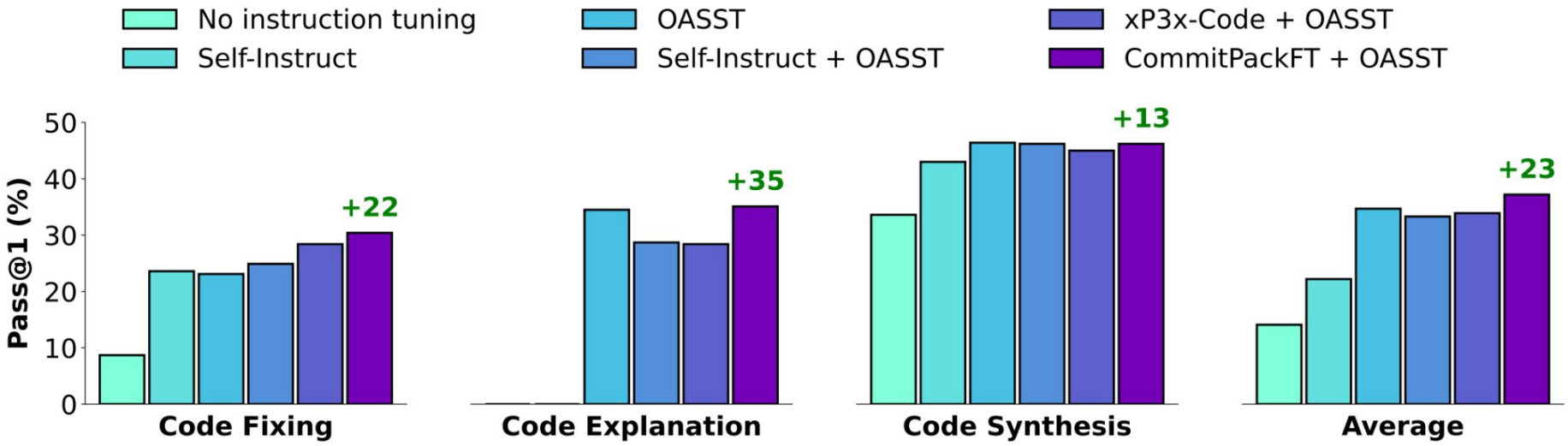


HumanEvalPack: Code Evaluation



HumanEvalPack: Code Evaluation

Benchmarking code instruction datasets on **HumanEvalPack** (Py) by instruction tuning StarCoder:



CommitPackFT + conversational data (OASST) leads to the highest average performance.

OctoCoder: Instruction Code Model

Model (↓)	Python	JavaScript	Java	Go	C++	Rust	Avg.
HUMANEVALFIX							
Non-permissive models							
InstructCodeT5+†	2.7	1.2	4.3	2.1	0.2	0.5	1.8
WizardCoder†	31.8	29.5	30.7	30.4	18.7	13.0	25.7
GPT-4	47.0	48.2	50.0	50.6	47.6	43.3	47.8
Permissive models							
BLOOMZ	16.6	15.5	15.2	16.4	6.7	5.7	12.5
StarChat-β	18.1	18.1	24.1	18.1	8.2	3.6	11.2
CodeGeeX2*	15.9	14.7	18.0	13.6	4.3	6.1	12.1
StarCoder	8.7	15.7	13.3	20.1	15.6	6.7	13.4
OCTOGEEX*	28.1	27.7	30.4	27.6	22.9	9.6	24.4
OCTOCODER	**30.4**	**28.4**	**30.6**	**30.2**	**26.1**	**16.5**	**27.0**

OctoCoder: Instruction Code Model

Model (↓)	Python	JavaScript	Java	Go	C++	Rust	Avg.
HUMANEVALEXPLAIN							
Non-permissive models							
InstructCodeT5+†	20.8	0.0	0.0	0.0	0.1	0.0	3.5
WizardCoder†	32.5	33.0	27.4	26.7	28.2	16.9	27.5
GPT-4	64.6	57.3	51.2	58.5	38.4	42.7	52.1
Permissive models							
BLOOMZ	14.7	8.8	12.1	8.5	0.6	0.0	7.5
StarChat-β	25.4	21.5	24.5	18.4	17.6	13.2	20.1
CodeGeeX2*	0.0	0.0	0.0	0.0	0.0	0.0	0.0
StarCoder	0.0	0.0	0.0	0.0	0.0	0.0	0.0
OCTOGEEX*	30.4	24.0	24.7	**21.7**	21.0	**15.9**	22.9
OCTOCODER	**35.1**	**24.5**	**27.3**	21.1	**24.1**	14.8	**24.5**

OctoCoder: Instruction Code Model

Model (↓)	Python	JavaScript	Java	Go	C++	Rust	Avg.
HUMANEVALSYNTHESIZE							
Non-permissive models							
InstructCodeT5+†	37.0	18.9	17.4	9.5	19.8	0.3	17.1
WizardCoder†	57.3	49.5	36.1	36.4	40.9	20.2	40.1
GPT-4	86.6	82.9	81.7	72.6	78.7	67.1	78.3
Permissive models							
BLOOMZ	15.6	14.8	18.4	8.4	6.5	5.5	11.5
StarChat-β	33.5	31.4	26.7	25.5	26.6	14.0	26.3
CodeGeeX2*	35.9	32.2	30.8	22.5	29.3	18.1	28.1
StarCoder	33.6	30.8	30.2	17.6	31.6	21.8	27.6
OCTOGEEX*	44.7	33.8	36.9	21.9	32.3	15.7	30.9
OCTOCODER	**46.2**	**39.2**	**38.2**	**30.4**	**35.6**	**23.4**	**35.5**

Thanks!

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, Shayne Longpre

OctoPack: Instruction Tuning Code Large Language Models

We thank Hugging Face for providing compute instances. We are extremely grateful to Rodrigo Garcia for the Rust translations, Dimitry Ageev and Calum Bird for help with GPT-4 evaluation, Loubna Ben Allal for help on evaluation, Arjun Guha for insightful discussions on chaining evaluation tasks to avoid evaluating with BLEU, Lewis Tunstall for help on the OASST data, Victor Sanh and Nadav Timor for discussions, Jiaxi Yang for logo editing and domain classification prompting design, Ghosal et al. (2023); Zeng et al. (2023) for design inspiration, Harm de Vries for feedback and all members of BigCode for general support. Finally, we thank every programmer who takes the time to write informative commit messages.