



Open and responsible development of
Large Language Models for code

Leandro von Werra | Machine learning engineer @ Hugging Face | lvwerra

Today's talk

BigCode Project

- 1 The Stack
- 2 StarCoder
- 3 Scaling LLMs

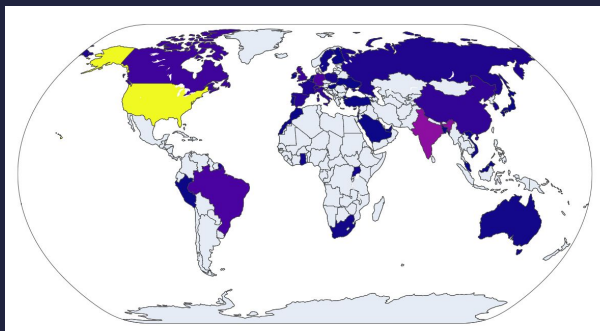
The BigCode community



BigCode: open-scientific collaboration

We are building LLMs for code in a collaborative way:

- 500+ participants
- 30+ countries



STARCODER:

MAY THE SOURCE BE WITH YOU!

Raymond Li² Loubna Ben Allal¹ Yangtian Zi⁴ Niklas Muennighoff¹ Denis Kocetkov²
Chenghao Mou⁵ Marc Marone⁸ Christopher Akiki^{9,10} Jia Li⁵ Jenny Chim¹¹
Qian Liu¹³ Evgenii Zheltonozhskii¹⁴ Terry Yue Zhuo^{15,16} Thomas Wang¹
Olivier Dehaene¹ Mishig Davaadorj¹ Joel Lamy-Poirier² João Monteiro²
Oleh Shliazhko² Nicolas Gontier² Nicholas Meade^{6,17} Armel Randy¹ Ming-Ho Yee⁴
Logesh Kumar Umapathi¹⁸ Jian Zhu¹⁹ Benjamin Lipkin²⁰ Muhtasham Oblokulov²¹
Zhiruo Wang⁷ Rudra Murthy²² Jason Stillerman²³ Siva Sankalp Patel²²
Dmitry Abulkhanov⁵ Marco Zocca²⁴ Manan Dey²⁵ Zhihan Zhang²⁶ Nour Fahmy²⁷
Urvashi Bhattacharyya²⁸ Suriya Gunasekar²⁹ Wenhao Yu²⁶ Swayam Singh³⁰
Sasha Luccioni¹ Paulo Villegas³¹ Maxim Kunakov³² Fedor Zhdanov³²
Manuel Romero⁵ Tony Lee³³ Nadav Timor³⁴ Jen Ding³⁵ Claire Schlesinger⁴
Hailey Schoelkopf³⁷ Jan Ebert³⁸ Tri Dao³³ Mayank Mishra²² Alex Gu²⁰
Jennifer Robinson³ Carolyn Jane Anderson³⁶ Brendan Dolan-Gavitt³⁹
Danish Contractor⁵ Siva Reddy^{2,6} Daniel Fried⁷ Dzmitry Bahdanau² Yacine Jernite¹
Carlos Muñoz Ferrandis¹ Sean Hughes³ Thomas Wolf¹ Arjun Guha^{4,12}
Leandro von Werra^{1,*} Harm de Vries^{2,*}

¹Hugging Face ²ServiceNow Research ³ServiceNow ⁴Northeastern University ⁵Independent
⁶Mila ⁷Carnegie Mellon University ⁸Johns Hopkins University ⁹Leipzig University
¹⁰ScaDS.AI ¹¹Queen Mary University of London ¹²Roblox ¹³Sea AI Lab ¹⁴Technion –
Israel Institute of Technology ¹⁵Monash University ¹⁶CSIRO's Data61 ¹⁷McGill University
¹⁸Saama AI Research Lab ¹⁹University of British Columbia ²⁰MIT ²¹Technical University of
Munich ²²IBM Research ²³University of Vermont ²⁴UnfoldML ²⁵SAP ²⁶University of
Notre Dame ²⁷Columbia University ²⁸Discover Dollar Pvt Ltd ²⁹Microsoft Research
³⁰University of Allahabad ³¹Telefonica I+D ³²Toloka ³³Stanford University ³⁴Weizmann
Institute of Science ³⁵The Alan Turing Institute ³⁶Wellesley College ³⁷Eleuther AI
³⁸Forschungszentrum Jülich ³⁹NYU

Corresponding authors (*) can be contacted at contact@bigcode-project.org

Developing Code LLMs is not only a technical problem!

- Consent of data subjects

- Can you use third-party copyrighted data to train machine learning models?

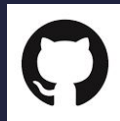
- Privacy risks

- Do LLMs distribute personally identifiable information without consent?

- Software safety and security

- Code LLMs may be used to generate malware or may provide code suggestions that are less secure

Scraped public
data sources



More info on the [Governance Card](#)

Closed development of LLMs

- Training data is not disclosed:
 - Content creators don't know if their data is used and there's no way to remove it
 - Limits scientific reproducibility
 - Potential benchmark contamination
- Model only available through API, which limits research on:
 - Safety and alignment
 - The model's inner workings (i.e. representations)
 - Adaptation methods like LoRA, and continuous prompt-tuning

Open & Responsible Research on LLMs

Open-access datasets

Permissive licensing

Data inspection

Opt-out available

PII removal

Attribution

Responsible use

Open-access models

Model weights available

Fine-tuning scripts

Low-precision inference

Reproducible research

Data preprocessing scripts

Model training framework

R&D notebooks

Evaluation Harness

Documentation

Dataset cards

Model cards

Governance card

Intellectual property

Code of conduct

OpenRAIL licenses

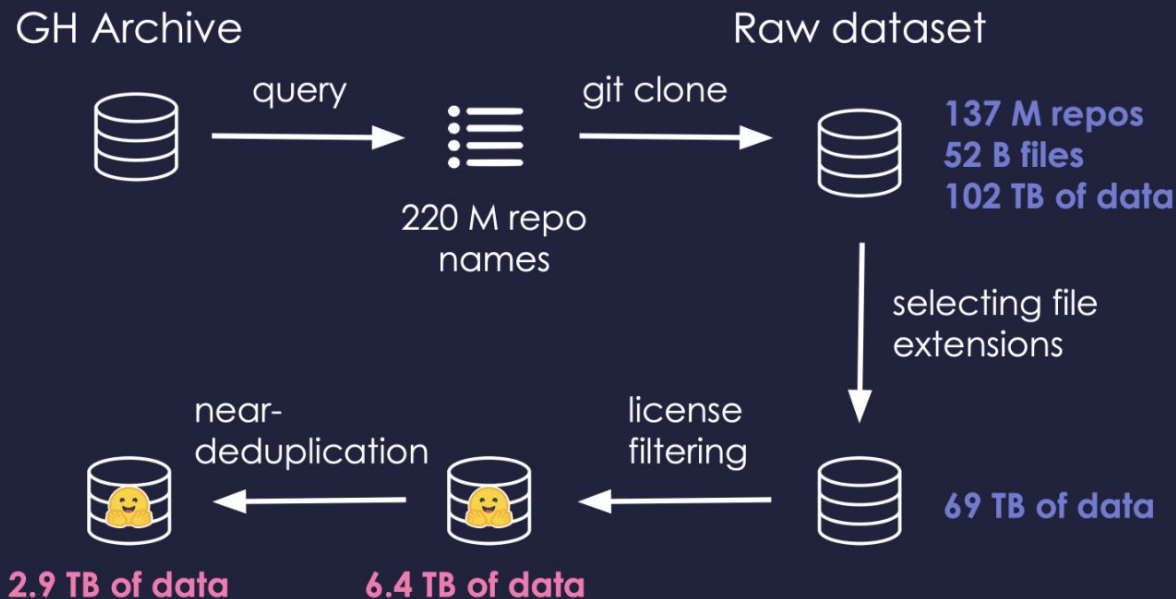
The Stack



The Stack

A dataset with **6.4TB** of **permissively licensed** source code in **358 programming languages** with a **data inspection** tool and **opt-out** mechanism

Data Collection



Find the filtered and deduplicated datasets at: www.hf.co/bigcode

Am I In The Stack?



BigCode

The Stack is an open governance interface between the AI community and the open source community.

Am I in The Stack?

As part of the BigCode project, we released and maintain [The Stack](#), a 3.1 TB dataset of permissively licensed source code in 30 programming languages. One of our goals in this project is to give people agency over their source code by letting them decide whether or not it should be used to develop and evaluate machine learning models, as we acknowledge that not all developers may wish to have their data used for that purpose.

This tool lets you check if a repository under a given username is part of The Stack dataset. Would you like to have your data removed from future versions of The Stack? You can opt-out following the instructions [here](#).

The Stack version:

v1.1

Your GitHub username:

Check!

Opt-out

Yes, there is code from 5 repositories in The Stack:

lvwerra/datasets

lvwerra/jupyterplot

lvwerra/pandas-profiling

lvwerra/transformers

lvwerra/trl

Opt-out

If you want your data to be removed from the stack and model training open an issue with [this link](#) (if the link doesn't work try right a right click and open it in a new tab) or visit <https://github.com/bigcode-project/opt-out-v2/issues/new?&template=opt-out-request.md>.

<https://huggingface.co/spaces/bigcode/in-the-stack>

Feedback from the opt-out form

- *"It should be opt-in instead of opt-out"*
- *"It is unfair to use my code without compensation"*
- *"There's PII in my code and I don't want it to be publicly exposed"*
- *"My code is of poor quality and unsuitable for training your AI model"*
- *"I am not confident about the current state of AI code generation. I am concerned that the generated code could be traced back to me and I'm held liable for issues in that code."*

Jennifer Ding's community research: it's both **better to know** AND **better to have a choice**.



StarCoder

Training Data - The Stack

- **Strong near-deduplication**

- language agnostic and shows consistent performance gains

- **Language selection**

- 86 languages out of 358 (popular languages, excluding configs)
- Includes GitHub issues, git commits, structured Jupyter notebooks

- **Data quality inspection**

- inspect 100 samples per source and derive filtering heuristics

- **Removing PII**

- StarPII: Model to detect/remove names, emails, password and keys

Architecture choices

What do people want from a code model?

- Easy downstream use
 - **15B parameters with code optimizations**
- Cheap generations
 - **Multi-Query Attention for reduced memory footprint**
- Long context
 - **Flash Attention to scale to 8,192 tokens context**
- Bi-directional context
 - **Fill-in-the-middle training objective**

Training setup

Infrastructure: 512 GPUs

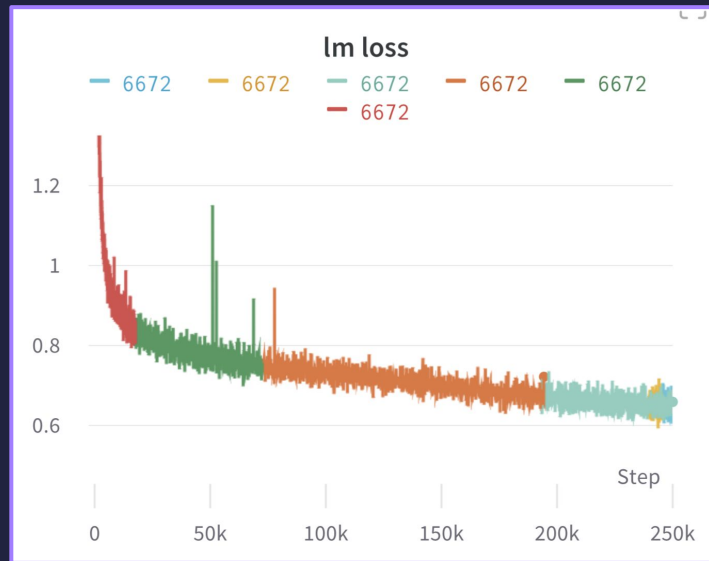
Model Distribution: TP=4, PP=4, DP=32

Batch size: 4M tokens
(or 512 at 8,192 sequence length)

Training length: 1T tokens / 250k steps

Training time: 24 days

Tool: Megatron-LM (w/ MQA + FlashAttn)
(<https://github.com/bigcode-project/Megatron-LM>)



“smooth sailing”



StarCoderBase

Trained on 1T tokens in
80+ languages

StarCoder

Additionally trained on
35B Python tokens

Evaluation - HumanEval/MBPP

Common observed failure mode:

solution here

Prompt fix:

<filename>solutions/solutions_1.py

Here is the correct implementation of the code exercise

Didn't observe similar gains for

CodeGen-Mono or StarCoderBase

Model	HumanEval	MBPP
LLaMA-7B	10.5	17.7
LaMDA-137B	14.0	14.8
LLaMA-13B	15.8	22.0
CodeGen-16B-Multi	18.3	20.9
LLaMA-33B	21.7	30.2
CodeGeeX	22.9	24.4
LLaMA-65B	23.7	37.7
PaLM-540B	26.2	36.8
CodeGen-16B-Mono	29.3	35.3
StarCoderBase	30.4	49.0
code-cushman-001	33.5	45.9
StarCoder	33.6	52.7
StarCoder-Prompted	40.8	49.5

VSCode extension

Auto-complete

```
Users > swayam > Desktop > Python main.py > ...  
1 def is_prime(num):  
2     return False  
  
def is_prime(num):  
    if num == 2:  
        return True  
    if num % 2 == 0:  
        return False  
    for i in range(3, num, 2):  
        if num % i == 0:  
            return False
```

<https://marketplace.visualstudio.com/items?itemName=HuggingFace.huggingface-vscode>

Membership test

```
Users > swayam > Desktop > Python main.py > is_prime  
1 def is_prime(num):  
2     return False  
3  
4 def is_prime(num):  
5     if num == 2:  
6         return True  
7     if num % 2 == 0:  
8         return False  
9     for i in range(3, num, 2):  
10        if num % i == 0:  
11            return False
```

Highlighted code was found in the stack.

Source: HF Code Autocomplete (Extension)

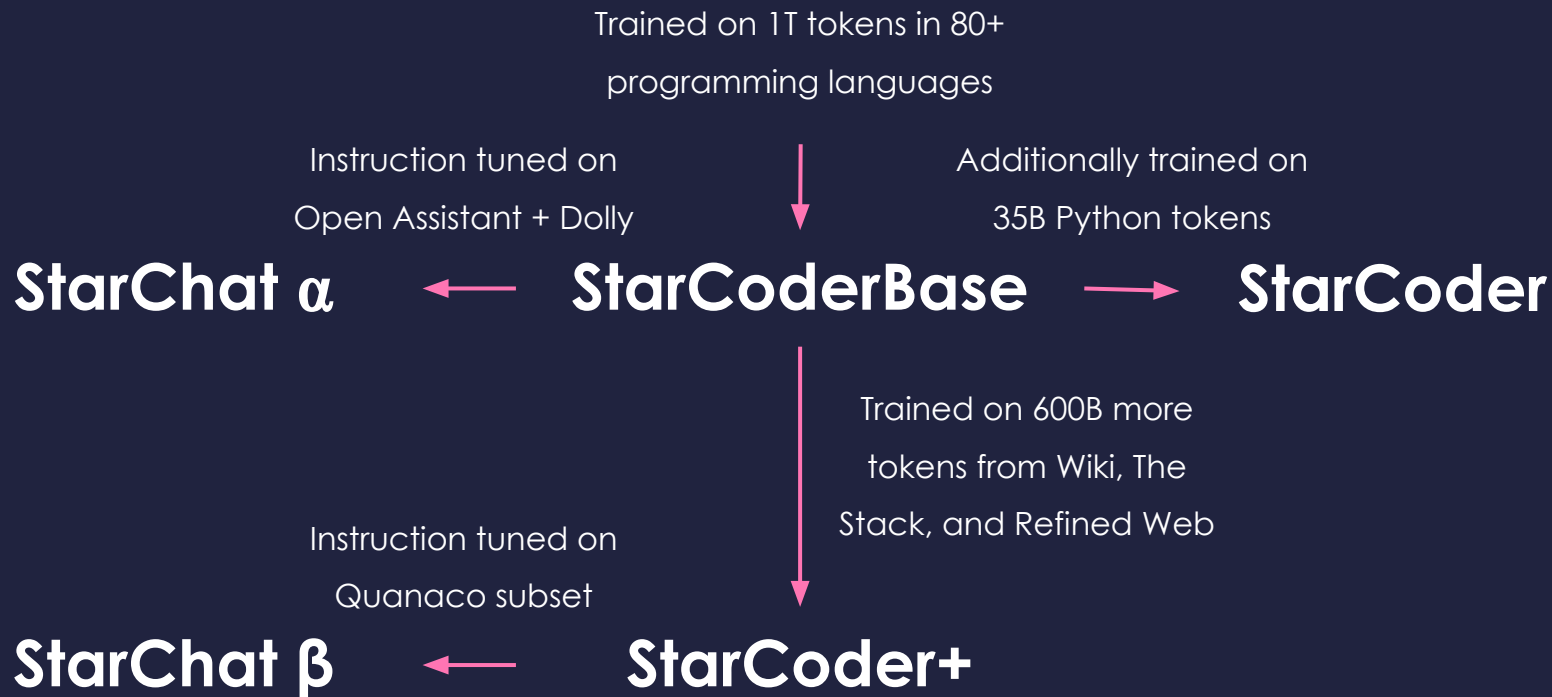
Go to stack search

Plugins and Demos

- VSCode plugin
- Jupyter plugin
- VIM plugin
- Emacs plugin
- VSCode + JetBrains plugins via refact.ai



Family Tree



Prompting StarCoder

```
1 Below are a series of dialogues between various people and
  an AI technical assistant. The assistant tries to be
  helpful, polite, honest, sophisticated, emotionally aware,
  and humble-but-knowledgeable. The assistant is happy to help
  with code questions, and will do its best to understand
  exactly what is needed. It also tries to avoid giving false
  or misleading information, and it caveats when it isn't
  entirely sure about the right answer. That said, the
  assistant is practical and really does its best, and doesn't
  let caution get too much in the way of being useful.

2
3 -----
4
5 Human: Write a function that takes two lists and returns a
  list that has alternating elements from each input list.
6
7 Assistant: Sure. Here is a function that does that.
8
9 ...
10 def alternating(list1, list2):
11     results = []
12     for i in range(len(list1)):
13         results.append(list1[i])
14         results.append(list2[i])
15     return results
16 ...
17
18 Human: Can you write some test cases for this function?
19
20 Assistant: Sure, here are some tests.
21
22 ...
23 assert alternating([10, 20, 30], [1, 2, 3]) == [10, 1, 20,
24 2, 30, 3]
25 assert alternating([True, False], [4, 5]) == [True, 4,
26 False, 5]
27 assert alternating([], []) == []
28 ...
29
30 Human: Modify the function so that it returns all input
  elements when the lists have uneven length. The elements
  from the longer list should be at the end.
31
32 Assistant: Here is the modified function.
```

excerpt from TA prompt

Inspired by Anthropic's HHH prompt: the TA prompt can make the model act as a tech assistant!

Can you give me Python code that tells me the weather in the city Bern?

Sure. Here is a simple script that uses the OpenWeatherMap API to retrieve data about the weather forecast in Bern.

```
import requests
import json
api_key = "<KEY>"
url = "http://api.openweathermap.org/data/2.5/forecast"
params = {"q": "Bern", "units": "metric", "APPID": api_key }
response = requests.get(url, params=params)
json_data = response.json()
print("Temperature:", round((json_data["main"]["temp"] - 273.15)))
print("Wind speed:", round(json_data["wind"]["speed"]))
print("Humidity:", round(json_data["main"]["humidity"], 1))
```



StarChat β

Write a function that counts the number of primes between 1 and 100.

Here is an example implementation in Python:

```
def count_primes(n):
    # Count all numbers up to n
    num = [True] * (n+1)

    # mark non-prime numbers as False
    for i in range(2, int(math.sqrt(n))+1):
        if (num[i]):
            for j in range(i*i, n + 1, i):
                num[j] = False

    return sum(num[:])

print("Number of prime numbers between 1 and 100:",
      count_primes(100))
# Output: Number of prime numbers between 1 and 100: 25
```

This code uses a list `num` with size equal to `n+1` where each element represents whether the corresponding index is a prime or not. We start by marking all indices from 2 onwards as `True` which means they are considered potential primes. Then we loop through all possible values of `i` such that square root of `n` is greater than or equal to `i`. For every `i`, we check if it's marked as `true`, meaning there could be a prime factor at this position. If so, we then mark all its multiples as `false` since these cannot be primes. Finally, we simply add together all the `truthy` elements in our list to get the total count of primes.

```
import math

def count_primes(n):
    # Count all numbers up to n
    num = [True] * (n+1)

    # mark non-prime numbers as False
    for i in range(2, int(math.sqrt(n))+1):
        if (num[i]):
            for j in range(i*i, n + 1, i):
                num[j] = False

    return sum(num[:])

print("Number of prime numbers between 1 and 100:",
      count_primes(100))
```

Number of prime numbers between 1 and 100: 27

CodeML OpenRAIL-M v1

- Royalty free access and use of the model
- Enables downstream distribution of derivatives
- Includes ethical use restrictions based on technical capabilities of the Model (e.g malware generation)
- Improved OpenRAIL license for commercialization

The Chinchilla trap and Harm's law



Scaling laws

How to distribute your compute budget between model parameters and training tokens?

Kaplan scaling laws

model 5.5x, data 1.8x

GPT-3

175B params, 300B tokens

Chinchilla scaling laws

model 3.2x, data 3.2x

Chinchilla

65B params, 1.6T tokens

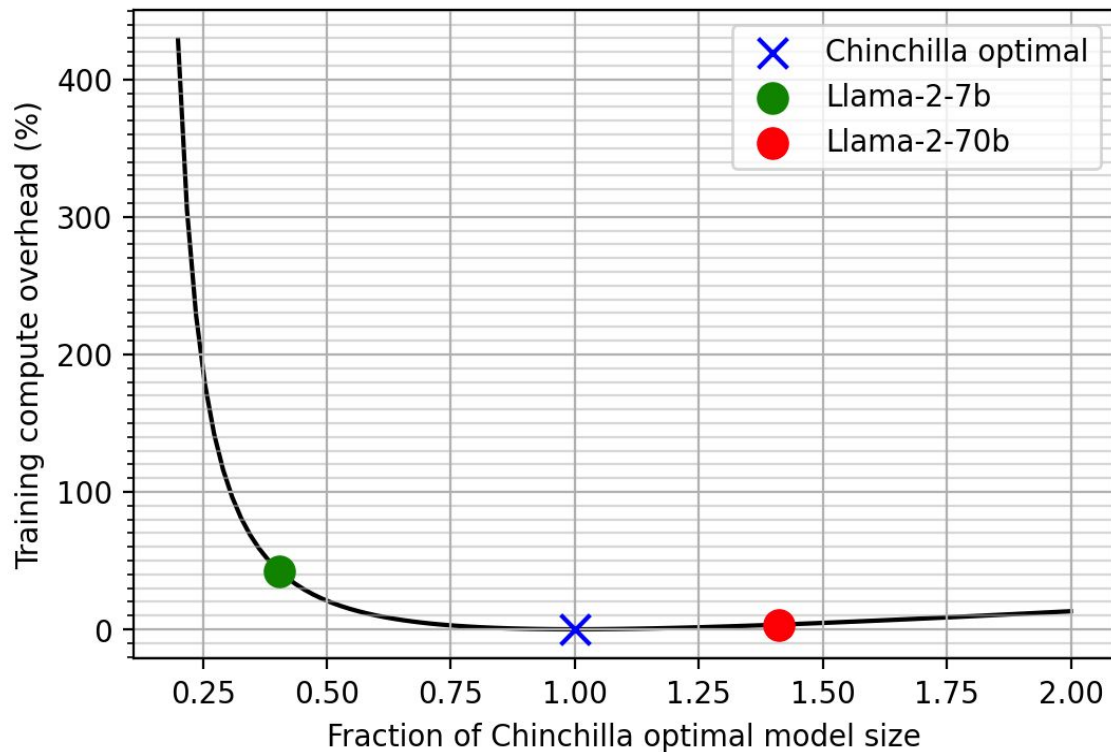
**Compute-optimal
≠ optimal**

LLaMA

7B params, 1T tokens



The compute vs model-size curve



Thank you! And come join us!



Questions?

www.bigcode-project.org

hf.co/bigcode

Jupyter

Jupyter Extension

A simple but efficient non-linear denoising estimator is obtained by thresholding the coefficients of f in a well chosen orthogonal basis $\mathcal{B} = \{\psi_m\}_m$ of \mathbb{R}^N .

In the following, we will focus on a wavelet basis, which is efficient to denoise piecewise regular images.

The hard thresholding operator with threshold $T \geq 0$ applied to some image f is defined as

$$S_T^0(f) = \sum_{| \langle f, \psi_m \rangle | > T} \langle f, \psi_m \rangle \psi_m = \sum_m s_T^0(\langle f, \psi_m \rangle) \psi_m$$

where the hard thresholding operator is

$$s_T^0(a) = \begin{cases} \alpha & \text{if } |\alpha| > T, \\ 0 & \text{otherwise.} \end{cases}$$

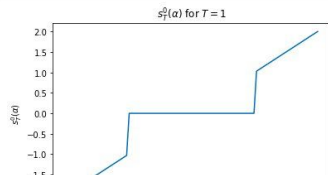
The denoising estimator is then defined as

$$\tilde{f} = S_T^0(f).$$

```
In [7]: # this code is generated by bigcode
import matplotlib.pyplot as plt
import numpy as np

def s_T_0(alpha, T=1):
    return np.where(np.abs(alpha) > T, alpha, 0)

plt.figure()
plt.plot(np.linspace(-2, 2, 100), s_T_0(np.linspace(-2, 2, 100), T=1))
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$s_T^0(\alpha)$')
plt.title(r'$s_T^0(\alpha)$ for $T=1$')
plt.show()
```



StarCoder as interpreter

Model input:

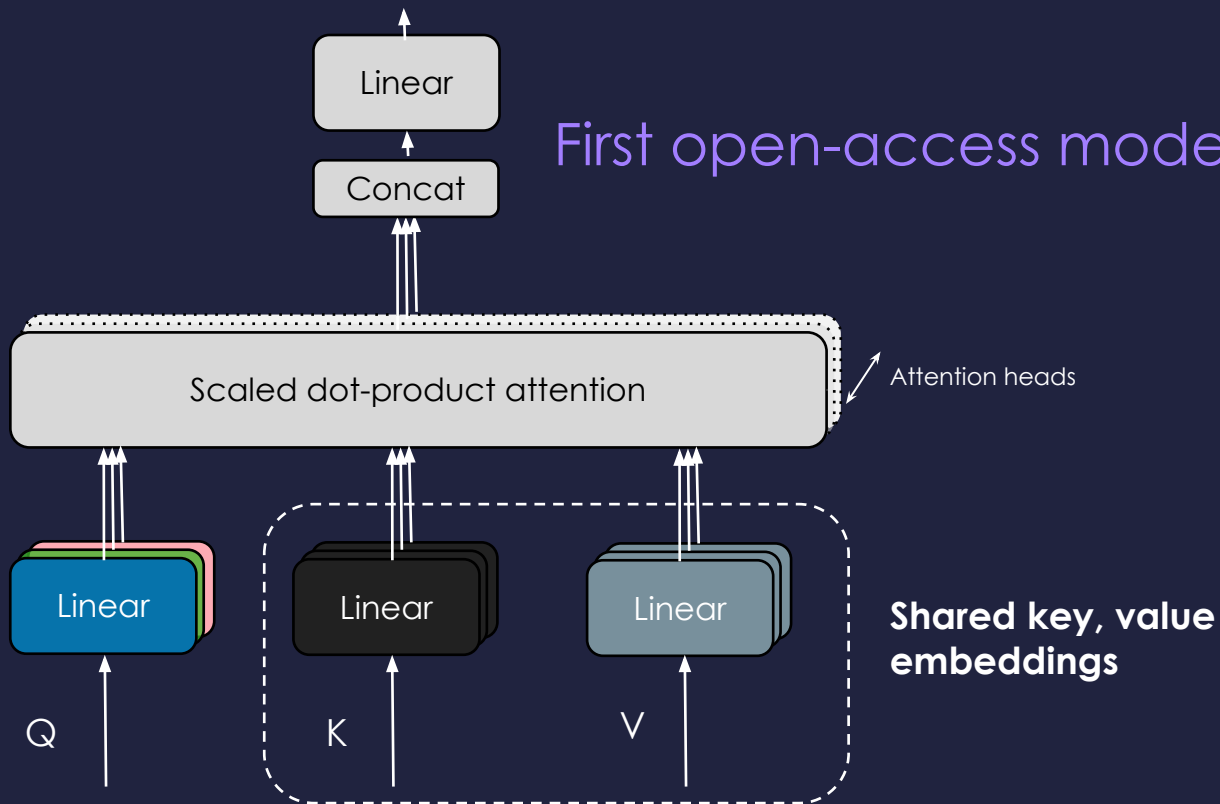
```
<jupyter_code>numbers = [1, 9, 8, 3, 27]
print([n*2 for n in numbers])<jupyter_output>
```

Model output:

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

<https://chrome.google.com/webstore/detail/jupyter-coder/kbedojkmlknhepcaggkdimefcbiifkjf>

Architecture choices: MQA



Evaluation - MultiPL-E

MultiPL-E contains translated versions of HumanEval

Language	CodeGen-16B-Multi	CodeGeeX	code-cushman-001	StarCoder	StarCoderBase
cpp	21.00	16.87	30.59	31.55	30.56
c-sharp	8.24	8.49	22.06	21.01	20.56
d	7.68	9.15	6.73	13.57	10.01
go	13.54	11.04	19.68	17.61	21.47
java	22.20	19.14	31.90	30.22	28.53
julia	0.00	0.29	1.54	23.02	21.09
javascript	19.15	16.92	31.27	30.79	31.70
lua	8.50	10.96	26.24	23.89	26.61
php	8.37	13.51	28.94	26.08	26.75
perl	3.42	8.09	19.29	17.34	16.32
python	19.26	21.62	30.71	33.57	30.35
r	6.45	3.92	10.99	15.50	10.18
ruby	0.00	3.34	28.63	1.24	17.25
racket	0.66	3.31	7.05	0.07	11.77
rust	4.21	7.88	25.22	21.84	24.46
scala	2.37	8.95	27.62	27.61	28.79
bash	0.61	2.75	11.74	10.46	11.02
swift	1.25	7.26	22.12	22.74	16.74
typescript	20.07	10.11	31.26	32.29	32.15

- outperforming open-access models
- competitive with code-cushman-001
- strong performance after fine-tuning
- big difference on some (ruby, r, swift)