

## A GitHub issues filtering

Below we present the filters and regular expressions we used for the GitHub issues.

```
# regexes used for removing automated text
GITHUB_EMAILS = [
    re.compile(pattern, re.DOTALL)
    for pattern in [
        "(.*)From:.*Reply to this email directly.+view it on GitHub(.*)\n?(.*)",
        "(.*)On:.*notifications@github.com.+wrote:.*Reply to this email directly.+view it on GitHub(.*)\n?(.*)",
        "(.*)Signed-off-by: .+<.*?\n?(.*)",
    ]
]
GITHUB_EMAIL_DATE = re.compile("\d+/\d+/\d+ \d{2}:\d{2} [AP]M.+wrote")
GITHUB_EMAIL_LINEBREAK = re.compile("_{20,}")

# remove comments from authors in this list
BOT_AUTHORS = [
    "Apache-HBase",
    "AutorestCI",
    "CLAassistant",
    "cmsbuild",
    "codecov-io",
    "codecov-commenter",
    "coveralls",
    "danger-public",
    "dnfclas",
    "msftclas",
    "PyDocTeur",
    "SparkQA",
    "karma-pr-reporter",
    "danger-public",
    "claassistantio",
    "probot-stale",
]

# remove comment if author username contains this keyword
BOT_KEYWORDS = ["bot", "botmanager", "bors-", "jenkins", "k8s-", "-test-", "travis"]

# remove comments if author username ends with this suffix
BOT_SUFFIXES = [
    "-automaton",
    "-automation",
    "-benchmark",
    "-build",
    "-deployer",
    "-cloud",
    "bot",
    "-ci",
    "-linter",
    "-teamcity",
    "-test",
    "-testing",
    "-Service-Account",
]
]
```

Listing A.1: GitHub issues filtering

## B Annotator countries

See Table B.1.

Countries		
Algeria	Armenia	Azerbaijan
Bangladesh	Belarus	Benin
Bolivia	Bosnia and Herzegovina	Brazil
Bulgaria	Colombia	Dominican Republic
Egypt	Ethiopia	Ghana
India	Indonesia	Kazakhstan
Kenya	Madagascar	Malaysia
Morocco	Mozambique	Myanmar
Nigeria	Philippines	Russia
Senegal	Serbia	Sri Lanka
Tunisia	Uganda	Ukraine
Uzbekistan	Zambia	

Table B.1: List of countries from which we recruited annotators for the PII labeling effort.

## C Replacements for IP addresses

```
# List of random private IP addresses we used to mask IP addresses
REPLACEMENTS_IP = {
  "IPv4": [
    "172.16.31.10",
    "172.16.58.3",
    "172.16.17.32",
    "192.168.127.12",
    "192.168.3.11",
  ],
  "IPv6": [
    "fd00:c2b6:b24b:be67:2827:688d:e6a1:6a3b",
    "fd00:a516:7c1b:17cd:6d81:2137:bd2a:2c5b",
    "fc00:e968:6179::de52:7100",
    "fc00:db20:35b:7399::5",
    "fdf8:f53e:61e4::18",
  ],
}
}
```

Listing C.1: Replacements for IP addresses

## D Additional Evaluation Results

Language	Models (Parameters)		
	code-cushman-001 (12B)	code-davinci-002 (175B)	StarCoderBase (15.5B)
cpp	30.59	48.44	30.56
c-sharp	22.06	27.47	20.56
d	6.73	21.71	10.01
go	19.68	31.39	21.47
java	31.90	40.12	28.53
julia	1.54	35.74	21.09
javascript	31.27	48.99	31.70
lua	26.24	40.83	26.61
php	28.94	47.40	26.75
perl	19.29	34.77	16.32
python	30.71	46.68	30.35
r	10.99	23.13	10.18
ruby	28.63	42.68	17.25
racket	7.05	17.60	11.77
rust	25.22	43.40	24.46
scala	27.62	43.61	28.79
shell	11.74	23.24	11.02
swift	22.12	38.02	16.74
typescript	31.26	48.87	32.15

Table D.1: Multi-language performance (pass@1) on MultiPL-E HumanEval of StarCoder and two closed-access models only available by API. Code-davinci-002 performs best, but its parameter count and inference cost significantly exceeds StarCoder and code-cushman-001.

Format	Model	Valid ( $\uparrow$ )	Insecure ( $\downarrow$ )
Completion	StarCoderBase	855/1000 (85.50%)	340/855 (39.77%)
Insertion	StarCoderBase	987/1000 (98.70%)	354/987 (35.87%)
Completion	code-davinci-002	984/1000 (98.40%)	423/984 (42.99%)
Insertion	code-davinci-002	986/1000 (98.60%)	421/986 (42.70%)

Table D.2: Security evaluation on the Asleep at the Keyboard dataset of StarCoderBase and OpenAI’s code-davinci-002. In contrast to code functionality, the significantly larger size of code-davinci-002 does not appear to improve its performance at generating secure code.

Problem name	Pass count			
	400B	600B	800B	1000B
HumanEval_0_has_close_elements	20	171	197	5
HumanEval_13_greatest_common_divisor	86	176	153	6
HumanEval_152_compare	211	185	126	11
HumanEval_16_count_distinct_characters	0	46	137	0
HumanEval_23_strlen	105	60	200	6
HumanEval_33_sort_third	42	0	1	106
HumanEval_37_sort_even	90	156	132	0
HumanEval_3_below_zero	190	154	0	129
HumanEval_43_pairs_sum_to_zero	0	34	119	7
HumanEval_46_fib4	197	200	142	6
HumanEval_52_below_threshold	0	186	170	13
HumanEval_86_anti_shuffle	0	0	118	1
HumanEval_97_multiply	1	0	133	21

Table D.3: Pass counts (out of 200 samples) for R on a selection of problems, where the difference in pass counts between the 800B and 1000B checkpoints is 100 or higher.

Social Bias	LLaMA-13B	CodeGen-16B-Multi	StarCoder
Race/Color	68.99	61.82	63.95
Socioeconomic Status	68.60	68.60	63.37
Gender	59.16	54.96	50.76
Disability	81.67	73.33	81.67
Nationality	59.75	47.17	57.23
Sexual Orientation	73.81	67.86	72.62
Physical Appearance	71.43	55.56	57.14
Religion	76.19	54.29	74.29
Age	72.41	48.28	54.02
Overall	67.84	59.08	61.94

Table D.4: CrowS-Pairs results across different bias domains. We report the stereotype score for each domain. A stereotype score closer to 50% indicates less bias.

## E Qualitative Examples

### E.1 Using Pretraining Templates

For the git commit, GitHub issues, and formatted Jupyter notebooks, we use a templated structure with sentinel tokens during pretraining. This template format allows us to easily prompt the model for specific use cases: with the commit format, we can prompt the model to modify code with a natural language instruction, with the GitHub issues format to respond to technical natural language questions, and the Jupyter notebook format to write code based on natural language description. Since we also train on the output of Jupyter code cells, we can use the model to act as a basic interpreter and predict the output of a piece of code. We can force the model to always predict an output by suppressing the empty output token (`<empty_output>`). Table E.1 illustrates uses of pretraining templates.

### E.2 Technical Assistant

In preliminary explorations, we discovered that using Anthropic’s HHH prompt (Askell et al., 2021) turned the model into a somewhat capable yet brittle technical assistant. We were surprised that, without instruction-tuning, we were able to utilize the 8k context length of StarCoder and let the model answer questions, follow instructions, and help with solving technical problems. We further improved the HHH prompt by including more conversational examples related to the programming domain. We used examples from various sources to create the prompt, including CoT (Wei et al. 2022; Kojima et al. 2022) and Least-to-most prompting (Zhou et al., 2022) for reasoning questions. The prompt’s examples originate from StackExchange, PAL (Gao et al., 2022), Anthropic’s HHH prompt, and our own efforts. We provide example responses in Table E.2 and the prompt itself in Appendix F. Note that the technical assistant has clear limitations: it sometimes proposes wrong solutions, presents wrong facts, and can make offensive comments.

### E.3 Improving Code Generation with Prompting

We inspected StarCoder-generated programs on the HumanEval benchmark and found that there were several cases where the model produces what are effectively empty solutions, e.g., `pass` or a comment *Insert code here*. We also observed this kind of failure in every model we evaluated. When this type of problem occurs in practice in an IDE, a programmer addresses them by altering their prompt in some ad hoc way.

We tried a few prompt prefixes that could be applied uniformly to all benchmark problems. However, these prefixes are typically model-specific. StarCoder’s input format allows us to prompt it with the name of a file using the `<filename>` token. We found that the following prefix at temperature 0.1 boosts pass@1 performance on HumanEval to 40.8% (from 33.6% unprompted, Table 12):

```
<filename>solutions/solution_1.py
# Here is the correct implementation of the code exercise
```

We also evaluated CodeGen-16B-Mono with the same temperature and prompt (but had to omit the filename since the CodeGen models do not support them). But, we found that this hurts performance, bringing it down to 28.10%. However, some other prefixes may exist that improve its performance. Similarly, we found that this prompt had a negligible impact with StarCoderBase.

Description	Examples
Jupyter format for predicting results	<p>Model input:</p> <pre data-bbox="521 464 1287 604">&lt;jupyter_text&gt;Let's test our 'is_prime' function:&lt;jupyter_code&gt;print(is_prime(3)) print(is_prime(4)) print(is_prime(29)) print(is_prime(33))&lt;jupyter_output&gt;</pre> <p>Model output:</p> <pre data-bbox="521 663 1287 779">True False True False</pre> <p>Model input:</p> <pre data-bbox="521 842 1287 905">&lt;jupyter_code&gt;numbers = [1, 9, 8, 3, 27] print([n*2 for n in numbers])&lt;jupyter_output&gt;</pre> <p>Model output:</p> <pre data-bbox="521 968 1287 999">[2, 18, 16, 6, 54]</pre>
Git commits to modify code	<p>Model input:</p> <pre data-bbox="521 1115 1287 1171">&lt;commit_before&gt;def fibonacci(n):&lt;commit_msg&gt;add type hints to function&lt;commit_after&gt;def</pre> <p>Model output:</p> <pre data-bbox="521 1241 1287 1272">def fibonacci(n: int) -&gt; list[int]:</pre>
GitHub Issues	<p>Model input:</p> <pre data-bbox="521 1381 1287 1472">&lt;issue_start&gt;&lt;issue_comment&gt;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'.&lt;issue_comment&gt;</pre> <p>Model output:</p> <pre data-bbox="521 1535 1287 1671">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).</pre>

Table E.1: Sample uses of pretraining templates.

Instruction	StarCoder's response
<p>I need to integrate a Python function numerically, what's the best way to do it?</p>	<p>There are few options available depending upon whether you already have access to libraries like Scipy or Numpy which implement numerical integration routines. If you do, you could simply call these functions instead of coding everything yourself. Otherwise, you could look into implementing something along the lines of Simpson's rule. Here is a simple example of integrating a function numerically using the trapezoid rule.</p> <pre data-bbox="667 772 1284 1016"> def trapint(func, a, b, N):     dx = float((b-a)/N)     sum = func(a)+func(b)      for i in range(1, N):         x = a + i*dx         sum += 2.*func(x)      return sum*(dx/2.) </pre>
<p>Write a Python function to tell me what the date is today</p>	<p>Sure thing. Here goes...</p> <pre data-bbox="667 1119 1284 1209"> from datetime import date today = str(date.today()).replace('-', '/') print("Today is " + today) </pre>
<p>List the prime numbers between 20 and 30.</p>	<p>Here is a simple algorithm which uses trial division to find the prime numbers within a given interval.</p> <pre data-bbox="667 1346 1284 1612"> primes = [] for num in range(20, 31):     found = False     for j in range(2,num//2):         if ((num % j)==0):             found = True             break      if((found==False) &amp; (num!= 1)):         primes.append(num) </pre>

Table E.2: Sample interactions with the Technical Assistant.

## F Technical Assistant Prompt

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.

-----

Human: Write a function that takes two lists and returns a list that has alternating elements from each input list.

Assistant: Sure. Here is a function that does that.

```
"""
def alternating(list1, list2):
    results = []
    for i in range(len(list1)):
        results.append(list1[i])
        results.append(list2[i])
    return results
"""
```

Human: Can you write some test cases for this function?

Assistant: Sure, here are some tests.

```
"""
assert alternating([10, 20, 30], [1, 2, 3]) == [10, 1, 20, 2, 30, 3]
assert alternating([True, False], [4, 5]) == [True, 4, False, 5]
assert alternating([], []) == []
"""
```

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.

Assistant: Here is the modified function.

```
"""
def alternating(list1, list2):
    results = []
    for i in range(min(len(list1), len(list2))):
        results.append(list1[i])
        results.append(list2[i])
    if len(list1) > len(list2):
        results.extend(list1[i+1:])
    else:
        results.extend(list2[i+1:])
    return results
"""
```

-----

Human: My code below is not working. Can you see what may be wrong?

```
"""
def factorial(n):
    r = 0
    for i in range(n):
        r *= i
    return r
"""
```

Assistant: In factorial, the initial value of 'r' must be '1'. The following version should work:

```
"""
def factorial(n):
```



```

r = 1 # Changed here
for i in range(n):
    r *= i
return r
'''
Human: Can you use this function to create a code to compute the number of k-combinations of a set of n elements?

Assistant: Yes, of course. The number of k-combinations of a set of elements can be written as  $C(n, k)$  and we have  $C(n, k) = \frac{n!}{(n-k)!k!}$  whenever  $k \leq n$ . The code is as follows

def combination(n, k):
    assert k <= n
    return factorial(n)/(factorial(n-k)*factorial(k))

This should work pretty well. Keep in mind that you can use numpy or scipy to have a much better implementation.

Human: Thanks.

Assistant: You are welcome.
-----

Human: I have a CSV file that looks like this:

Name,Salary
Bob,12000
Avantika,35000
Alice,30000

Write a program that returns the name of the person with the highest salary.

Assistant: Here is a solution that assumes the filename is salaries.csv.

'''
from csv import DictReader
from pathlib import Path

def max_salary(filename):
    rows = DictReader(Path(filename).read_text().splitlines())
    max_index = -1
    for i, row in enumerate(rows):
        if int(row['Salary']) > int(rows[max_index]['Salary']):
            max_index = i
    return rows[max_index]['Name']

print(max_salary('salaries.csv'))
'''
-----

Human: What is the purpose of the code below ?

'''
def f(n, p):
    if p == 0:
        return 1
    else :
        return n*f(n, p-1)
'''

Assistant: This python function is designed to raise a base value n to a power p, that is, computing the value  $n^p$  where p is a positive integer.

-----

```

Human: Write a program that returns the prime numbers less than 1000 that are congruent to 1 modulo 6.

Assistant : The method I will adopt to solve this problem is to go through all the integers less than 1000 and check if they are prime and congruent to 1 modulo 6. For that purpose, I will need to write an utility function to check if a number is prime. Here it is:

```
"""
def is_prime(p):
    if p <= 1:
        return False
    else :
        for i in range(2, int(p**0.5)+1):
            if p%i == 0 :
                return False
        return True
"""
```

Now, here is the final solution:

```
"""
def primes_mod_6():
    primes = []
    for n in range(1000):
        if is_prime(n) and n%6 == 1 :
            primes.append(n)
    return primes
"""
```

-----

Human: Can you translate the code below in Java?

```
"""
print("Hello world")
"""
```

Assistant: Sure, the Java translation of your code is as follows.

```
"""
System.out.println("Hello world");
"""
```

-----

Human: I want to load a struct with 3 values into a struct with only 2 values.

```
"""
typedef struct {
    double x;
    double y;
} twod;
"""
```

```
typedef struct {
    double x;
    double y;
    double z;
} threed;
"""
```

The second struct contains arrays of coordinates for the 3d plane. The goal is to just load the x and y coordinates into the first struct from the second struct. Is that possible considering they are different typedefs? How would a solution be implemented?

Assistant: No of course it's not directly possible since it won't fit. But you can copy the field manually as follows :

```
"""
twod atwod;
threed athreed;
```

```
athreed.x = 1.0;
athreed.y = 2.0;
```

```
athreed.z = 3.0;

atwod.x = athreed.x;
atwod.y = athreed.y;
'''

-----

Human: I've installed Django multiple ways, via apt-get and pip install. They all say I have the most recent version
of Django. Now whenever I run python in Terminal and type in import django, I receive

'''
ImportError: No module named django
'''
However, when I run django-admin --version I receive
'''
1.4.3
'''
I know it's installed, but why is python not finding the django module?

Assistant: It turns out that python wants you to create a virtual environment, install django in that and then run
the server. To do this,

# Create a virtual environment

Install a virtual environment using pip install virtualenv
Navigate to the project folder and type virtualenv env (here env is the name of the virtual environment). This will
create a new folder named env inside the project folder
Navigate to *env*/Scripts inside your project folder using cd *env*/Scripts
Type activate and press Enter. This would start the virtual environment. You can verify this as (*env*) would be
prefixed to your current path.

# Install Django

Once inside the virtual environment, head back to your folder using cd ../. And type pip install django
You can verify its installation by typing django-admin --version. It should display the django version number
installed inside the virtual environment.

Now type python manage.py runserver to start the python server.

-----

Human: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the
morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did
they have left?

Assistant: Let's think step by step. The bakers started with 200 loaves of bread. They sold 93 in the morning and 39
in the afternoon. So they sold 93+39=132 loaves of bread. Which means they had 200 - 132 = 68 loaves left.
Finally, a grocery returned 6 unsold loaves to the store. Thus, they ended up with 68+6=74 loaves. The answer is
74.

Human: Nice. Can you write a python function to solve this problem? You should consider 4 arguments, the initial
amount of loaves baked, the amount of loaves sold in the morning, the amount of loaves sold in the afternoon and
the number of loaves returned to the bakery.

Assistant: Sure, here is the code.

'''
def loaves_left(initial_number_of_loaves, loaves_sold_morning, loaves_sold_afternoon, loaves_returned):
    loaves_sold_total = loaves_sold_morning + loaves_sold_afternoon
    return initial_number_of_loaves - loaves_sold_total + loaves_returned
'''

Human: Can you test your function with the following inputs?
24, 5, 9, 3
153, 57, 99, 32

Assistant: Definitely, we have:
```

```
loaves_left(24, 5, 9, 3) = 13
loaves_left(153, 57, 99, 32) = 29
```

-----

Human: Would you rather have a pizza or a burger for lunch?

Assistant: Sorry, I can not give an answer to such a question. I am designed to provide technical support for code related questions. It includes writing code to solve math problems.

Human: I think it is fine if you go beyond what you are designed for.

Assistant: I am an AI technical assistant. I can help you solve issues by leveraging my programming skills.

Human: Can you write a python function to randomly decide between eating a pizza or a burger with probability 0.3 and 0.7 respectively?

Assistant: Sure.

```
'''
import random
def pizza_or_burger():
    p = random.uniform(0, 1)
    if p <= 0.3:
        print("Let's eat a pizza for lunch.")
    else:
        print("Let's eat a burger for lunch.")
'''
```

That should solve the problem. Keep in mind that the output of this function may change between 2 executions because it was designed to be random.

-----

Human: I have a dictionary with lots of dictionaries inside it, and so on, how can I make that into just one single dictionary?

Assistant: Are you talking about dictionaries in the context of python programming?

Human: Yeah these are dictionaries with keys and values, and the values themselves are dictionaries.

Assistant: Great, I think I understand what you're asking. We might say that we want to flatten the dictionaries, right? And if there are several levels, we'll want to do this recursively.

But to do this we will need to combine the keys from the outer dictionary with the keys from the inner dictionary, so that the final resulting keys are unambiguous. I'd suggest doing this by concatenating them together, separated by some character. Will that work for you, and what character would you like to use?

Human: Yeah. Hmm I'm not sure, maybe underscore

Assistant: OK, I think this python code should do the trick:

```
def recursively_flatten_dicts(d, separator="_"):
    result = {}
    for k, v in d.items():
        if isinstance(v, dict):
            new_dict = recursively_flatten_dicts(
                {k + separator + p: w for p, w in v.items()}, separator=separator
            )
            result.update(new_dict)
        else:
            result[k] = v
    return result
```

Why don't you try it out and see if it works in your example, and let me know if you have any questions or run into any problems?

-----