

**Dataset Card**

- **Name:** InfiBench
- **Description:** Evaluation Dataset for the Question-Answering Capabilities of Code Large Language Models
- **URL:** <https://infi-coder.github.io/infibench> (all resources) / <https://huggingface.co/datasets/llylly001/InfiBench> (data part)
- **Version:** 2.1
- **License:** Creative Commons Attribution Share Alike 4.0
- **Citation:**

```
@misc{infibench,
  title={InfiBench: Evaluating the Question-Answering Capabilities
of Code Large Language Models},
  howpublished = "\url{https://infi-coder.github.io/infibench}",
  author={InfiBench},
  year={2024}
}
```

- **DOI:** doi:10.57967/hf/2474

- **Responsible AI — Data Collection:**

Data source is downloaded from the publicly available StackExchange archive (<https://archive.org/download/stackexchange>, [https://ia904700.us.archive.org/view\\_archive.php?archive=/6/items/stackexchange/stackoverflow.com-Posts.7z](https://ia904700.us.archive.org/view_archive.php?archive=/6/items/stackexchange/stackoverflow.com-Posts.7z)). Especially, we use the preprocessed version from <https://huggingface.co/datasets/mikex86/stackoverflow-posts> where all posts are formatted in Markdown text.

We choose to keep only the questions with at least three positively voted answers and an officially accepted answer, which turn out to be 1,090,238 questions. For these one million questions, we choose to keep frequently viewed and relatively new questions.

Utilizing the mandatory question tags of these questions, we then manually construct a tag tree that covers the 200 most frequent tags, enabling us to identify the top programming languages and areas for 14,330105 out of these 17,402 questions. We exclude 6 programming languages that either describe data or are domain-specific: JSON, regex, Markdown, YAML, CSV, and SQL. As a result, we compile 13,854 questions that serve as the initial seed set.

We randomly sample from the initial seed set. Then we recruited five domain experts inside our company to create the benchmark from the sampled initial seed set, each in charge of one area. The annotation process is composed of three steps: (1) Question Selection and Type Annotation; (2) Prompt Paraphrasing. (3) Correctness Criterion Annotation.

- **Responsible AI — Data Biases:**

The data essentially serves as an evaluation benchmark. We foresee data biases in the following aspects:

- (1) Non-standard evaluation. Alongside the data is a comprehensive benchmark of existing code LLMs. The benchmark scores are evaluated under a specific set of hyperparameters (e.g. temperature 0.2, top probability 0.9, best@10 at question level). Data usage under different evaluation conditions may result in misleading comparison results and conclusions.
- (2) Usage misinterpretation. The benchmark focuses on evaluating the response correctness of code LLMs for a set of real-world developers' questions. Our evaluation standard does not specifically take other aspects (naturalness, conciseness, fairness, politeness, etc) into consideration. Hence, this is risk of overinterpreting the evaluation results. When evaluating a code LLM, we recommend combining this benchmark score with other evaluations to be a more comprehensive evaluation.

(3) Potential data contamination. Though we have made our efforts to reduce the impact of data contamination, future code LLMs may train or fine-tune on this benchmark dataset to improve the score on InfiBench. This could be challenging to prevent as a cost of being fully public. On the other hand, as responsible LLM developers, we hope future practitioners would report how they use the benchmark data if beyond the original scope (for evaluation use).

- **Responsible AI — Personal Sensitive Information:** During the data construction process, our domain experts paraphrased the question prompts to remove personal and sensitive information (PII) and a cross validation stage was introduced to further ensure the PII removal.

507

508 **Croissant Dataset Description:** [https://huggingface.co/datasets/llylly001/](https://huggingface.co/datasets/llylly001/InfiBench/blob/main/croissant-infibench.json)  
509 [InfiBench/blob/main/croissant-infibench.json](https://huggingface.co/datasets/llylly001/InfiBench/blob/main/croissant-infibench.json). Note that the Croissant format is  
510 mainly designed for machine learning dataset description. However, InfiBench is more than a dataset;  
511 it is an evaluation benchmark including response evaluation standards, tools, and an accompanying  
512 leaderboard. Hence, the Croissant script records only the CSV file and covers question prompts  
513 and evaluation standards; whereas the open-source evaluation tool and leaderboard are not recorded  
514 which can be separately downloaded from <https://infi-coder.github.io/infibench>.

515 **Data Accessibility.** As briefly mentioned in the main text, all materials are made publicly available  
516 and accessible at the website: <https://infi-coder.github.io/infibench> without personal  
517 request. The materials include three parts: (1) Benchmark questions and evaluation metrics —  
518 this part is additionally uploaded to Hugging Face (URL and DOI are in the above dataset card).  
519 (2) Automatic evaluation tool — this part is uploaded and maintained in a dedicated GitHub repo  
520 <https://github.com/infi-coder/infibench-evaluator>. In addition, we uploaded our ex-  
521 tension of `bigcode-evaluation-harness` [5], namely `infibench-evaluation-harness` to a  
522 dedicated GitHub repo <https://github.com/infi-coder/infibench-evaluation-harness>.  
523 The extension includes the inference code on InfiBench for all evaluated LLMs. (3) Evaluation  
524 raw data and leaderboard — the leaderboard is displayed on the website [https://infi-coder.](https://infi-coder.github.io/infibench)  
525 [github.io/infibench](https://infi-coder.github.io/infibench) and the raw model responses are stored in the website repo <https://github.com/infi-coder/infibench>. All materials are under the Creative Commons At-  
526 tribution Share Alike 4.0 license. In the above dataset card and Appendix B, we anticipate potential  
527 inappropriate usage of the benchmark and we encourage the practitioners to document their usage  
528 of the benchmark if beyond model evaluation. In the future, we will continue the maintenance and  
529 expansion of the benchmark. Furthermore, we are developing an adaptor for automatic evaluation on  
530 Hugging Face so that InfiBench can be integrated into the Hugging Face Open LLM Leaderboard [4]  
531 to further ease the evaluation burden.  
532

## 533 B Limitations, Societal Impacts, and Future Work

534 In this appendix, we expand our discussion of limitations, potential societal impacts, and future work.

535 **Evaluation Metric.** In InfiBench, the expert-annotated evaluation metric is designed to mainly  
536 focus on response correctness, more specifically, whether the response contains key information  
537 that solves the given question. Concretely, the metric may evaluate whether the response passes a  
538 given set of unit tests, whether it suggests the right API or concept, whether it follows the instruction  
539 to provide relevant information, etc. Hence, the score comes with two limitations: (1) The score  
540 is subjective since the metric is annotated by human experts without an explicit and universal  
541 standard. Note that we did not aim to provide an objective metric since the developers’ views of  
542 response correctness intrinsically vary and diverge for these diverse questions. On the other hand, we  
543 introduce a cross-validation and calibration stage to improve the metric representativeness of most  
544 developers’ standards. We leave it as a future work to further quantitatively measure and improve  
545 the metric representativeness. (2) The score focuses mainly on correctness. Several other aspects

546 define a model’s usability, such as language naturalness (including conciseness, politeness, etc),  
547 trustworthiness (refusal of risky questions, fairness, unbiasedness, privacy, etc), and system-level  
548 metrics (latency, throughput, parallelism-friendliness, etc). Model evaluators and practitioners may  
549 keep in mind that InfiBench score is not a comprehensive usability measurement of code LLMs, and  
550 we strongly encourage them to combine InfiBench score with benchmarks on these other aspects (c.f.  
551 [6, 38]) to comprehensively evaluate LLMs.

552 **Data Contamination.** The limitations and mitigations on data contamination are discussed in  
553 Section 2.6. In addition, as a side effect of open source, future code LLMs may leverage the  
554 benchmark data to deliberately introduce data contamination to achieve a high score in InfiBench. To  
555 partly detect such data contamination, our evaluation of using the original stack Overflow answers  
556 might be a proxy. According to Table 4(a), even gold extraction from human answers cannot saturate  
557 the benchmark while strong LLMs like GPT-4 surpassed human answers. Hence, if a future model  
558 achieves scores close to human answers (between 50% and 65%) but cannot further improve beyond  
559 human along with scaling, data contamination may potentially happen. Detecting data contamination  
560 is itself a research topic where research on member inference attacks [33, 26] is involved. We did not  
561 integrate a detection module in the current release of InfiBench but we are planning to inspect this  
562 topic in the future.

563 **Labeling Cost.** InfiBench construction involves human labeling cost, where domain experts para-  
564 phrase the source question post and label the evaluation metric. Such a cost prevents the InfiBench  
565 from scaling up at the current stage. In attempts to mitigate this limitation, we explored a few  
566 alternative evaluation metrics such as dialogue similarity with officially accepted answers. However,  
567 these alternatives either require a language model which may induce bias and heavy computing  
568 cost, or deviate away from domain experts’ correctness judgment. We leave the exploration of more  
569 scalable metrics and annotation procedures as future work and make the benchmark fully open source  
570 so community involvement may boost the expansion.

## 571 C Difficulty Grouping

572 We systematically evaluated GPT-4 and GPT-3.5-turbo on the benchmark following the evaluation  
573 protocol in Section 3, based on which we classify the benchmark questions into five disjoint difficulty  
574 groups.

- 575 • Level 1 (93 questions, 39.7%): GPT-3.5-turbo can achieve a mean score  $\geq 0.5$ .
- 576 • Level 2 (55 questions, 23.5%): Among the rest questions, those where GPT-4’s mean score  $\geq 0.5$ .
- 577 • Level 3 (44 questions, 18.8%): Among the rest questions, those where GPT-4 with sampling  
578 temperature 1.0 can achieve a maximum score  $\geq 0.5$  among 10 trials.
- 579 • Level 4 (18 questions, 7.7%): Among the rest questions, those GPT-4 with sampling temperature  
580 0.2 can achieve a positive score among 100 trials.
- 581 • Level 5 (24 questions, 10.3%): The remaining questions, i.e., GPT-4 cannot get score among 100  
582 trials.

583 Appendix D shows each code LLM’s score in each difficulty group. The mean scores strictly decrease  
584 for higher difficulty levels, highlighting that the question difficulty is in general consistent across  
585 different code LLMs and our group assignment is reasonable. We hope that the grouping can help  
586 better reveal the strengths and weaknesses of a code LLM for different questions.

587 Question examples by difficulty groups are in Appendix H.

## 588 D Evaluation Details and Full Benchmark Results

589 **Evaluation Details of Code LLMs.** For proprietary model evaluation, we did not specify the max  
590 tokens to generate and found out that the longest response generated by GPT-4 has 662 tokens with  
591 Code Llama tokenizer.

592 For open-source model evaluation, for models with over 30B parameters, due to the GPU memory  
593 limit and efficiency concerns, we impose the longest context constraint of 4,096 tokens and experiment  
594 just once. Since there is only one question whose GPT-4 context (prompt + GPT-4 response) can  
595 exceed 4,096 tokens, we think this context constraint has little effect, reducing the score by 0.37% at  
596 most. For models within 30B parameters, since GPT-4 response has at most 662 tokens, we set the  
597 max number of tokens to generate to be  $\min\{1024, \text{context length} - \text{prompt length}\}$ , providing some  
598 wiggle room. Meanwhile, we repeat the evaluation three times for models within 30B parameters.

599 **Evaluation Details of Original Stack Overflow Answers.** As listed in Table 4(a) and Table 6,  
600 besides evaluating LLM responses, we evaluated the score of human-written original Stack Overflow  
601 answers since the question prompts are paraphrased from Stack Overflow. We consider three settings:  
602 (1) evaluating the officially-accepted answer post (note that we select only the Stack Overflow  
603 questions with an officially-accepted answer into the benchmark); (2) evaluating the highest-voted  
604 answer post (note that any registered user can equally vote for or against an answer); and (3) evaluating  
605 the highest-voted answer posts up to 10 and recording the highest score achieved by any post. For  
606 the last setting, we chose the number 10 because the main evaluation metric of model response is  
607  $\text{best@10}$ . Moreover, we observe that all officially accepted answers for InfiBench questions are  
608 among the top 10 highest-voted answer posts. Note that there is no randomness of scores from Stack  
609 Overflow answers, so we do not repeat the evaluation nor report the standard deviation.

610 As expected, the last setting achieves the highest score 65.18% among the three settings. Due  
611 to its consistency with models’ evaluation metric  $\text{best@10}$ , we deem this score most comparable  
612 with scores from LLMs. Interestingly, when considering only one answer post, the second setting,  
613 selecting the highest-voted answer, is better than the first setting, selecting the officially accepted  
614 answer.

615 **Full Benchmark Results.** We present the full leaderboard in Table 6 (by descending order of  
616 InfiBench scores) and Table 7 (by alphabetical order of model family names). These tables are  
617 expanded from the aggregated Table 4. In these tables, we show model properties including size  
618 and context length. We also present HumanEval [3] scores since HumanEval is one of the most  
619 widely used benchmarks for evaluating code LLMs (further discussion in Appendix E). Furthermore,  
620 we represent the score breakdown by difficulty levels, problem types, and evaluation metric types.  
621 The proportion of each difficulty level can be found in Appendix C, and the proportion of each  
622 problem type and evaluation metric type is shown in Table 3(a,b). InfiBench score can be computed  
623 by the weighted sum of breakdown subscores by proportions. We present the score of human-written  
624 original Stack Overflow answers in the last three rows.

625 In tables, the mean scores are computed from scores of all 106 code LLMs. We observe that the  
626 mean overall score, 37.82%, is still much inferior to human answers (which achieves over 50% even  
627 with just one attempt). The model performance is monotonically decreasing for higher difficulty  
628 levels; relatively equivalent across different problem types; and weaker under blank-filling and  
629 dialogue-similarity metrics than keyword-matching and unit-testing metrics.

## 630 E Additional Findings and Discussion

631 In this appendix, we present additional findings and discussion that are omitted from Section 3.

### 632 E.1 Correlations between InfiBench and HumanEval Scores

633 We study the correlation between InfiBench and HumanEval  $\text{pass@1}$  scores for different LLMs. In  
634 Figure 5, we plot LLMs with both InfiBench and HumanEval scores, in total 66 LLMs, in Table 6 as  
635 a scatter plot. The figure shows that scores on the two benchmarks are generally positively correlated,  
636 with a Pearson correlation coefficient  $r = 0.8058$ . If conducting a linear regression, we would  
637 observe that different model types (i.e., general/code model, base/finetuned model) share almost the  
638 same linear relationship, indicating that both benchmarks can reflect the model capability in general.

Table 6: Full leaderboard of all benchmarked LLMs ranked by InfBench scores. Evaluation protocol in Section 3 and details explained in Appendix D. Icon “

Rank	Model Family	Model Name	Size (# Params.)	Context Length	InfBench Score	HumanEval	Difficulty Levels					Problem Type			Evaluation Metric Type				
							Level 1	Level 2	Level 3	Level 4	Level 5	Code Completion	Code Debugging	Knowledge QA	Config & Debugging	KeyWord Matching	Unit Testing	Blank Filling	Dialogue Similarity
1	GPT-4	GPT-4-0613	?	8192	70.64% ± 0.82%	96.4	95.11%	92.88%	51.90%	31.91%	0.0%	95.2%	69.74%	68.5%	66.63%	66.61%	66.00%	58.08%	84.27%
2	GPT-4	GPT-4-turbo-1106	?	8192	68.42% ± 0.38%	85.4	89.90%	78.57%	54.16%	30.93%	16.20%	74.82%	65.36%	67.47%	62.98%	64.98%	67.40%	53.91%	52.85%
3	GPT-4	GPT-4o-2024-10-13	?	8192	66.19%	80.2	81.29%	74.66%	46.43%	28.05%	5.21%	69.0%	99.32%	65.65%	61.76%	61.59%	76.06%	49.14%	70.35%
4	Claude 3	Claude 3 Opus	?	200000	63.89%	70.3	84.36%	78.95%	39.88%	31.76%	18.06%	65.18%	62.94%	65.86%	60.49%	60.07%	61.89%	59.36%	44.91%
5	Mistral Open	CodeLlama-3B-Instruct	22B	32768	62.98% ± 0.56%	81.11	88.64%	69.90%	49.97%	17.11%	5.90%	68.7%	63.65%	61.07%	54.28%	57.72%	73.33%	45.92%	57.08%
6	DeepSeek Code	deepseek-coder-33b-instruct	33B	16384	62.56%	80.02	87.58%	72.02%	44.12%	15.83%	16.67%	71.26%	57.14%	63.14%	56.81%	59.01%	67.00%	30.00%	36.62%
7	Phind	Phind-CodeLlama-34B-v2	34B	4096	62.56%	80.02	87.58%	72.02%	44.12%	15.83%	16.67%	71.26%	57.14%	63.14%	56.81%	59.01%	67.00%	30.00%	36.62%
8	Phind	Phind-CodeLlama-34B-v1	34B	4096	58.47%	65.85	81.38%	63.85%	47.05%	22.63%	5.21%	66.13%	56.94%	56.79%	49.48%	53.17%	66.00%	38.78%	35.39%
9	Mistral	mistral-large	?	32768	60.5%	67.88	81.76%	66.99%	41.66%	23.62%	4.17%	66.69%	50.10%	60.21%	52.89%	55.71%	67.00%	45.64%	42.66%
10	Claude 3	Claude 3 Sonnet	?	200000	60.4%	74.9	80.13%	65.59%	42.48%	18.06%	15.28%	62.61%	52.34%	63.61%	52.12%	54.22%	66.00%	46.35%	25.62%
11	CodeLlama 3	CodeLlama-3B	?	200000	60.4%	74.9	80.13%	65.59%	42.48%	18.06%	15.28%	62.61%	52.34%	63.61%	52.12%	54.22%	66.00%	46.35%	25.62%
12	DeepSeek LLM	deepseek-llm-67b-chat	67B	4096	57.41%	68.48	82.96%	63.03%	39.09%	22.60%	5.21%	61.42%	52.73%	58.72%	55.63%	53.14%	63.00%	51.41%	36.68%
13	GPT-3.5	GPT-3.5-turbo-0613	?	4096	56.47% ± 1.34%	72.6	93.08%	49.77%	31.36%	14.30%	7.64%	64.91%	48.50%	59.47%	49.64%	51.28%	70.07%	40.90%	40.13%
14	Mistral	mistral-small	?	32768	55.62% ± 0.46%	72.6	82.98%	55.98%	35.72%	22.58%	10.07%	63.56%	44.12%	64.13%	47.75%	50.56%	68.00%	39.08%	53.32%
15	Mistral Open	mistral-8x7b-instruct	46.7B / 12.9B	32768	55.55%	57.8	82.98%	56.72%	31.53%	24.00%	17.36%	61.0%	51.37%	60.89%	53.13%	55.14%	68.00%	35.88%	61.25%
16	Qwen	Qwen-72B	72B	32768	55.34%	57.8	81.98%	57.40%	41.61%	13.24%	4.17%	61.06%	53.16%	58.79%	44.03%	50.43%	64.00%	45.96%	36.41%
17	DeepSeek Code	deepseek-coder-6.7b-instruct	6.7B	16384	53.25% ± 0.40%	60.22	77.88%	56.30%	35.18%	18.89%	9.72%	65.9%	46.4%	52.46%	42.12%	48.24%	70.00%	26.90%	23.8%
18	Qwen	Qwen-72B Chat	72B	32768	52.97%	57.8	82.44%	47.09%	36.69%	18.34%	9.38%	58.67%	45.81%	60.12%	44.31%	49.26%	59.00%	43.08%	33.95%
19	MagiCoder	CodeLlama-34B-DS-6.7B	34B	4096	52.71% ± 0.17%	60.22	77.97%	50.42%	40.20%	13.64%	6.00%	61.39%	51.88%	58.97%	53.79%	53.29%	66.00%	35.36%	24.19%
20	WizardLM	WizardCoder-Python-34B-V1.0	34B	16384	52.59%	60.22	77.97%	50.42%	40.20%	13.64%	6.00%	61.39%	51.88%	58.97%	53.79%	53.29%	66.00%	35.36%	24.19%
21	Phind	Phind-CodeLlama-34B-Python-v1	34B	4096	52.17%	70.22	80.54%	48.44%	42.58%	8.57%	1.04%	54.41%	52.34%	57.11%	41.47%	51.04%	57.80%	27.91%	39.76%
22	MagiCoder	MagiCoder-DS-6.7B	6.7B	16384	51.46% ± 1.09%	60.22	77.97%	50.42%	28.91%	25.93%	6.48%	62.54%	46.4%	55.74%	43.44%	45.64%	69.13%	31.45%	27.86%
23	CodeLlama 3	CodeLlama-3B-Instruct	34B	16384	51.46% ± 1.09%	60.22	77.97%	50.42%	28.91%	25.93%	6.48%	62.54%	46.4%	55.74%	43.44%	45.64%	69.13%	31.45%	27.86%
24	o1	o1	?	4096	49.58%	50.79	72.60%	55.07%	33.16%	18.43%	9.72%	51.71%	48.37%	61.36%	37.04%	38.04%	51.20%	47.76%	28.55%
25	WizardLM	WizardCoder-Python-7B-V1.0	7B	16384	49.10% ± 1.59%	50.79	72.60%	55.07%	33.16%	18.43%	9.72%	51.71%	48.37%	61.36%	37.04%	38.04%	51.20%	47.76%	28.55%
26	WizardLM	WizardCoder-Python-13B-V1.0	13B	16384	48.99% ± 0.92%	50.79	72.60%	55.07%	33.16%	18.43%	9.72%	51.71%	48.37%	61.36%	37.04%	38.04%	51.20%	47.76%	28.55%
27	CodeLlama 3	CodeLlama-3B	34B	16384	47.36%	62.19	76.21%	46.76%	34.19%	16.17%	0.35%	52.69%	48.29%	50.67%	41.32%	48.71%	53.73%	20.45%	29.61%
28	CodeLlama 3	CodeLlama-13B-Instruct	13B	16384	46.37% ± 1.26%	62.19	76.21%	46.76%	34.19%	16.17%	0.35%	52.69%	48.29%	50.67%	41.32%	48.71%	53.73%	20.45%	29.61%
29	Zephyr	Zephyr-7B beta	7B	32768	46.31% ± 1.11%	62.19	76.21%	46.76%	34.19%	16.17%	0.35%	52.69%	48.29%	50.67%	41.32%	48.71%	53.73%	20.45%	29.61%
30	StarCoder2	15B-Instruct	15B	16384	45.89% ± 0.95%	60.77	70.37%	50.21%	24.15%	11.44%	6.83%	56.02%	38.52%	46.30%	38.56%	40.55%	60.07%	25.21%	45.01%
31	DeepSeek MoE	deepseek-moe-16b-chat	16B / 2.8B	16384	45.18% ± 1.85%	60.77	68.15%	46.72%	27.55%	10.17%	11.23%	47.19%	46.54%	45.88%	39.99%	45.71%	44.73%	25.85%	20.70%
32	OctoPack	Qwen-72B	72B	16384	45.18% ± 1.85%	60.77	68.15%	46.72%	27.55%	10.17%	11.23%	47.19%	46.54%	45.88%	39.99%	45.71%	44.73%	25.85%	20.70%
33	Qwen	Qwen-14B-Chat	14B	8192	43.69% ± 1.09%	45.3	67.61%	47.64%	21.87%	9.63%	7.52%	44.59%	42.15%	47.09%	39.99%	41.61%	44.30%	34.19%	28.21%
34	Qwen	Qwen-1.8B-Chat	1.8B	8192	43.49% ± 0.63%	45.3	68.91%	36.25%	27.73%	10.28%	15.39%	45.39%	42.12%	46.33%	38.48%	41.87%	42.73%	36.18%	34.79%
35	MagiCoder	MagiCoder-DS-6.7B	6.7B	16384	43.47% ± 0.21%	45.3	67.61%	48.33%	23.11%	13.64%	0.69%	52.77%	40.42%	48.14%	25.61%	38.37%	56.73%	29.81%	38.07%
36	CodeLlama 3	CodeLlama-3B	34B	16384	43.13%	45.3	66.02%	40.76%	36.69%	13.64%	0.00%	50.16%	40.88%	58.65%	54.13%	55.49%	43.07%	20.92%	39.29%
37	CodeLlama 3	CodeLlama-7B-Instruct	70B	4096	42.82%	45.3	59.08%	41.41%	38.48%	12.22%	7.64%	58.27%	44.99%	46.87%	42.38%	48.34%	32.00%	16.99%	5.62%
38	StarCoder2	15B	15B	16384	42.52% ± 1.24%	45.3	64.99%	41.67%	29.02%	13.77%	3.70%	47.00%	37.31%	46.76%	36.87%	43.86%	42.20%	18.44%	0.0%
39	MagiCoder	MagiCoder-CL-7B	7B	16384	41.71% ± 0.76%	45.3	67.61%	48.33%	23.06%	10.33%	0.35%	49.26%	35.11%	45.41%	33.47%	37.85%	52.27%	19.91%	39.21%
40	CodeLlama 3	CodeLlama-13B	13B	16384	41.66% ± 0.84%	45.3	62.77%	40.40%	31.11%	7.97%	7.41%	58.17%	44.56%	43.00%	41.72%	45.44%	34.20%	14.79%	4.4%
41	DeepSeek Code	deepseek-coder-1.8b-instruct	1.8B	16384	41.32% ± 1.12%	45.3	65.48%	41.42%	25.48%	6.89%	2.78%	41.91%	42.86%	41.89%	41.80%	42.36%	16.52%	24.32%	
42	CodeLlama 3	CodeLlama-13B-Python	13B	16384	41.31% ± 0.90%	45.3	62.93%	40.80%	28.61%	10.37%	5.21%	49.95%	44.69%	36.68%	27.22%	40.58%	51.07%	11.92%	13.64%
43	WizardLM	WizardCoder-15B-V1.0	15B	16384	41.01% ± 0.22%	45.3	66.19%	40.34%	21.72%	12.42%	1.74%	44.80%	34.54%	47.68%	35.29%	38.43%	47.60%	22.31%	35.01%
44	Mistral	mistral-medium	?	32768	40.95% ± 0.04%	45.3	72.99%	30.34%	19.14%	8.15%	7.29%	41.49%	34.99%	49.19%	39.99%	38.54%	42.67%	33.88%	18.26%
45	gemma	gemma-7b	7B	8192	40.68% ± 1.23%	45.3	60.94%	42.94%	28.86%	5.79%	4.86%	42.66%	36.7%	47.95%	34.52%	40.68%	41.8%	19.4%	30.48%
46	CodeLlama 3	CodeLlama-70B	70B	4096	40.60%	45.3	60.99%	37.42%	35.68%	7.59%	4.17%	47.18%	39.10%	39.09%	33.21%	40.89%	40.9%	19.23%	8.5%
47	CodeLlama 3	CodeLlama-70B-Python	70B	4096	40.29%	45.3	59.14%	36.07%	41.06%	7.59%	0.00%	42.03%	43.04%	46.02%	32.46%	41.78%	41.00%	10.96%	19.50%
48	OctoPack	OctoPack	6B	8192	40.14% ± 0.55%	45.3	62.54%	37.84%	26.39%	15.67%	2.20%	42.24%	33.23%	46.76%	31.10%	39.85%	39.96%	20.90%	31.11%
49	DeepSeek LLM	deepseek-llm-67b-base	67B	4096	39.87%	45.3	65.48%	41.42%	25.48%	6.89%	2.78%	41.91%	42.86%	41.89%	41.80%	42.36%	16.52%	24.32%	
50	Llama 2	Llama2-70B-Chat	70B	4096	39.30%	45.3	56.95%	38.02%	33.71%	7.96%	7.64%	35.65%	42.87%	42.56%	36.11%	40.89%	34.40%	22.44%	28.14%
51	DeepSeek Code	deepseek-coder-33b-base	33B	16384	38.75%	52.45	56.73%	44.55%	19.85%	14.95%	8.33%	33.36%	43.73%	46.06%	31.23%	43.99%	25.00%	14.49%	28.02%
52	o1	o1	?	4096	38.14% ± 0.88%	52.45	52.73%	38.20%	34.73%	12.53%	7.64%	33.36%	39.81%	42.54%	38.33%	43.26%	23.33%	15.32%	15.69%
53	Llama 2	Llama2-7B-Chat	7B	4096	37.06%	52.45	51.01%	38.20%	34.73%	12.53%	7.64%	33.36%	39.81%	42.54%	38.33%	43.26%	23.33%	15.32%	15.69%
54	CodeLlama 3	CodeLlama-7B	7B	16384	37.62% ±														

Table 7: Full leaderboard of all benchmarked LLMs by model family name for indexing. Same content as Table 6. Evaluation protocol in Section 3 and details explained in Appendix D. Icon “” stands for proprietary models otherwise open-source. As a reference, HumanEval scores digested from [22] and each model’s report are shown. Bar colors stand for General Base, General Finetuned, Code Base, and Code Finetuned models respectively. Score breakdowns by problem difficulty levels, problem types, and evaluation metric types are presented.

No	Model Family	Model Name	Size (Param.)	Context Length	InfBench Score	HumanEval	Difficulty Levels					Problem Type				Evaluation Metric Type			
							Level 1	Level 2	Level 3	Level 4	Level 5	Code Completion	Code Debugging	Knowledge QA	Config & Env Debugging	Keyword Matching	Unit Testing	Blank Filling	Dialogue Summary
1	01.AI	Yi-34B-Chat	34B	4096	38.14% ± 0.58%	/	76.81%	47.15%	28.32%	26.30%	4.17%	44.10%	44.75%	62.29%	49.84%	53.14%	35.40%	36.15%	33.97%
2	01.AI	Yi-8B-Chat	8B	4096	26.39% ± 0.42%	/	52.73%	38.20%	34.37%	12.53%	7.64%	33.36%	39.81%	42.54%	38.33%	43.26%	23.83%	15.32%	15.60%
3	01.AI	Yi-6B	9B	4096	26.39% ± 0.42%	39	41.18%	29.89%	14.75%	3.33%	0.00%	20.83%	27.06%	34.48%	24.58%	30.21%	17.60%	5.96%	14.34%
4	01.AI	Yi-34B	34B	4096	22.01%	/	34.64%	26.46%	7.73%	1.85%	4.17%	23.15%	16.96%	31.36%	23.10%	12.40%	6.15%	11.46%	
5	01.AI	Yi-6B	6B	4096	19.97% ± 1.24%	/	31.84%	18.91%	13.15%	0.99%	2.78%	13.75%	23.72%	23.54%	20.37%	23.42%	14.58%	0.00%	4.54%
6	Baichuan2	Baichuan2-13B-Chat	13B	4096	24.40% ± 1.34%	17.5	33.77%	27.47%	24.19%	6.85%	14.12%	37.03%	35.97%	36.39%	24.88%	34.62%	31.07%	22.83%	18.28%
7	Baichuan2	Baichuan2-7B-Chat	7B	4096	26.32% ± 1.23%	/	42.14%	28.83%	16.84%	3.55%	5.56%	29.02%	26.36%	32.91%	19.63%	28.30%	27.40%	3.65%	49.66%
8	Baichuan2	Baichuan2-7B-Base	7B	4096	23.50% ± 1.56%	/	43.01%	21.48%	18.67%	6.47%	4.98%	22.46%	26.54%	31.79%	25.63%	30.05%	16.24%	9.23%	10.70%
9	ChatGLM	ChatGLM3-6B	6B	8192	28.23% ± 0.83%	52.4	36.59%	23.93%	13.01%	5.99%	4.17%	21.05%	22.93%	28.98%	21.49%	26.03%	19.33%	4.68%	10.70%
10	Claude3	Claude 3 Opus	? ?	200000	37.82% ± 0.83%	81.9	84.50%	78.55%	39.98%	31.76%	18.06%	65.14%	62.94%	63.86%	60.49%	60.07%	61.00%	59.36%	44.91%
11	Claude3	Claude 3 Sonnet	? ?	200000	35.20% ± 0.83%	74.9	80.13%	65.55%	42.48%	18.06%	15.28%	62.61%	62.34%	63.61%	52.12%	54.22%	66.89%	46.35%	25.22%
12	Claude3	Claude 3 Haiku	? ?	200000	32.70% ± 0.83%	75.9	79.86%	66.06%	40.23%	21.76%	10.42%	61.71%	48.68%	62.85%	56.71%	55.78%	58.40%	44.62%	26.40%
13	Code Llama	CodeLlama-34B-Instruct	34B	16384	40.40% ± 0.54%	78.9	72.60%	55.07%	33.16%	18.43%	9.72%	51.71%	48.37%	61.36%	57.04%	48.14%	51.20%	47.76%	38.58%
14	Code Llama	CodeLlama-7B	7B	16384	47.36%	45.11	47.07%	47.64%	29.37%	21.20%	13.54%	50.09%	51.25%	26.99%	41.85%	41.18%	37.37%	24.85%	
15	Code Llama	CodeLlama-13B-Instruct	13B	16384	46.37% ± 1.26%	50.6	60.97%	45.99%	34.37%	11.42%	7.52%	48.65%	45.18%	49.67%	39.83%	47.17%	50.47%	20.90%	12.47%
16	Code Llama	CodeLlama-34B-Python	34B	16384	43.13%	53.29	66.02%	40.76%	36.06%	6.94%	0.00%	50.14%	40.48%	43.64%	34.13%	40.10%	50.09%	27.63%	16.58%
17	Code Llama	CodeLlama-70B-Instruct	70B	4096	42.82%	45.8	59.08%	44.14%	38.48%	12.22%	7.64%	38.20%	44.99%	46.87%	42.38%	48.34%	32.00%	16.09%	5.62%
18	Code Llama	CodeLlama-13B	13B	16384	41.66% ± 0.84%	42.6	62.77%	40.76%	36.06%	6.94%	0.00%	45.62%	40.76%	43.64%	34.13%	40.10%	50.09%	27.63%	16.58%
19	Code Llama	CodeLlama-13B-Python	13B	16384	41.31% ± 0.90%	45.9	62.93%	40.40%	28.61%	10.37%	5.21%	49.95%	44.60%	36.68%	27.22%	40.58%	51.07%	11.92%	13.64%
20	Code Llama	CodeLlama-70B	70B	4096	40.60%	55.5	60.59%	37.42%	35.88%	7.59%	4.17%	47.18%	39.10%	39.09%	33.21%	40.54%	45.90%	19.23%	8.56%
21	Code Llama	CodeLlama-70B-Python	70B	4096	40.29%	58.9	59.14%	36.07%	41.06%	7.59%	0.00%	42.03%	43.04%	40.76%	32.46%	41.78%	41.00%	16.99%	19.50%
22	Code Llama	CodeLlama-7B	7B	20198	37.62% ± 1.29%	29.9	59.81%	29.08%	31.19%	7.97%	4.86%	38.17%	44.56%	43.06%	37.17%	28.41%	31.77%	28.15%	0.00%
23	Code Llama	CodeLlama-7B-Instruct	7B	16384	35.15% ± 1.02%	45.6	53.69%	35.79%	24.82%	7.59%	1.39%	36.46%	37.13%	35.00%	30.65%	35.97%	34.87%	15.77%	13.83%
24	Code Llama	CodeLlama-7B-Python	7B	16384	32.89% ± 0.45%	40.48	51.02%	28.69%	24.32%	7.59%	6.94%	30.38%	38.34%	32.37%	29.81%	35.27%	30.00%	8.97%	11.31%
25	CodeLlama	CodeLlama-7B-Chat	7B	16384	32.89% ± 0.45%	33.20	31.40%	47.14%	14.02%	2.22%	4.86%	18.78%	19.97%	25.39%	14.44%	22.08%	16.11%	4.10%	9.78%
26	CodeLlama	CodeLlama-7B-multi	6B	8192	19.88% ± 0.36%	39.46	20.79%	13.62% ± 1.18%	10.79%	2.22%	4.86%	10.79%	19.97%	25.39%	14.44%	22.08%	16.11%	4.10%	9.78%
27	CodeLlama	CodeLlama-7B	16B	2048	16.97% ± 1.11%	33.20	27.46%	18.30%	8.08%	1.23%	1.39%	13.00%	17.28%	24.01%	10.23%	20.77%	7.58%	7.05%	0.00%
28	CodeLlama	CodeLlama-7B-Instruct	7B	2048	29.57% ± 1.53%	31.2	50.36%	22.07%	20.25%	6.67%	0.46%	28.76%	25.96%	37.70%	25.77%	32.35%	24.76%	11.54%	0.00%
29	CodeLlama	CodeLlama-7B	? ?	16384	21.25% ± 1.17%	33.66	33.66%	19.26%	13.61%	5.27%	3.70%	15.05%	19.63%	30.5%	22.70%	25.42%	13.30%	2.61%	4.33%
30	DeepSeek	deepseek-coder-33b-instruct	33B	16384	40.80% ± 0.83%	80.02	87.58%	72.02%	44.12%	15.83%	16.67%	71.26%	57.14%	63.14%	56.81%	59.01%	77.00%	30.00%	36.09%
31	DeepSeek	deepseek-coder-13b-instruct	13B	16384	41.32% ± 1.12%	61.6	65.48%	41.42%	25.48%	6.30%	2.78%	41.91%	42.56%	42.88%	36.38%	41.80%	42.20%	16.52%	24.32%
32	DeepSeek	deepseek-coder-33b-base	33B	16384	38.75%	52.45	56.73%	44.55%	19.85%	14.95%	8.33%	33.36%	43.73%	46.06%	31.23%	43.99%	25.50%	14.49%	28.02%
33	DeepSeek	deepseek-coder-7b-base	7B	16384	33.66% ± 1.24%	45.83	53.26%	37.95%	14.02%	8.56%	2.78%	36.56%	32.40%	37.83%	25.00%	35.17%	33.47%	11.92%	8.81%
34	DeepSeek	deepseek-coder-7b-7nnaq-base	7B	16384	28.92% ± 1.12%	45.8	45.62%	33.11%	11.67%	4.54%	4.51%	26.82%	27.41%	37.87%	23.17%	30.64%	24.79%	10.64%	19.44%
35	DeepSeek	deepseek-coder-1.3b-base	1.3B	16384	23.17% ± 1.47%	32.13	37.06%	26.74%	8.03%	2.84%	4.17%	16.05%	20.93%	34.16%	24.68%	27.02%	14.40%	4.68%	24.92%
36	DeepSeek	deepseek-llm-67b-chat	67B	4096	40.96%	42.7	82.96%	63.03%	39.09%	22.60%	5.21%	61.42%	52.73%	58.75%	55.63%	53.14%	63.00%	51.41%	36.68%
37	DeepSeek	deepseek-llm-67b-base	67B	4096	39.87%	42.7	87.15%	48.73%	24.32%	9.17%	4.17%	35.50%	43.17%	46.15%	34.40%	39.98%	36.00%	30.00%	24.46%
38	DeepSeek	deepseek-llm-7b-chat	7B	4096	36.75% ± 1.80%	46.2	55.46%	39.38%	22.94%	6.30%	6.37%	34.08%	29.57%	46.76%	38.83%	39.15%	30.13%	15.90%	35.86%
39	DeepSeek	deepseek-llm-7b-base	7B	4096	25.34% ± 1.08%	42.7	36.58%	30.99%	15.33%	2.01%	5.66%	19.59%	27.42%	29.23%	27.22%	28.67%	15.60%	8.97%	25.29%
40	DeepSeek	deepseek-moe-16b-chat	16B / 2.8B	16384	45.18% ± 1.65%	42.7	68.15%	46.72%	27.55%	10.17%	11.23%	47.19%	46.54%	45.58%	39.09%	45.71%	44.73%	25.85%	20.70%
41	DeepSeek	deepseek-moe-16b-base	16B / 2.8B	16384	26.59% ± 0.97%	41.6	41.68%	31.71%	12.27%	4.21%	0.00%	28.09%	25.69%	31.15%	19.60%	27.77%	27.11%	5.38%	22.26%
42	gemma	gemma-7b	7B	8192	30.80% ± 0.83%	28.7	60.94%	42.94%	28.86%	5.75%	4.86%	42.60%	36.37%	47.75%	34.52%	40.68%	41.40%	19.84%	30.48%
43	gemma	gemma-2b-it	2B	8192	27.49% ± 0.52%	35.4	43.43%	29.33%	13.99%	0.62%	8.56%	22.98%	26.43%	38.79%	24.97%	29.17%	20.49%	6.40%	8.44%
44	gemma	gemma-7b	7B	8192	16.05% ± 1.07%	35.4	27.46%	14.76%	5.96%	2.65%	0.00%	6.98%	19.73%	15.00%	14.05%	19.37%	6.42%	2.56%	8.45%
45	gemma	gemma-2b	2B	8192	14.62% ± 0.50%	35.4	23.18%	13.23%	10.53%	4.07%	0.00%	12.16%	12.89%	24.70%	8.33%	16.99%	11.33%	0.00%	10.13%
46	GPT-3.5	GPT-3.5-turbo-0613	? ?	4096	36.04% ± 1.34%	72.6	93.08%	49.77%	31.56%	14.30%	7.64%	64.91%	48.50%	59.47%	49.64%	51.28%	70.07%	40.90%	40.00%
47	GPT-4	GPT-4-turbo-1106	? ?	8192	40.80% ± 0.83%	80.02	91.29%	82.88%	51.91%	0.00%	0.00%	66.61%	69.16%	68.55%	66.41%	76.00%	58.08%	48.76%	
48	GPT-4	GPT-4-turbo-1106	? ?	8192	68.42% ± 0.83%	85.4	80.90%	78.57%	54.16%	30.93%	16.20%	74.82%	65.36%	67.47%	62.98%	64.98%	76.40%	53.91%	52.85%
49	GPT-4	GPT-4o-2024-05-13	? ?	8192	60.19% ± 0.69%	90.0	91.29%	78.66%	46.43%	28.05%	5.21%	75.00%	59.32%	65.65%	61.70%	61.59%	76.03%	70.23%	
50	HiTianYuan	Yuan2-21B-HF	21B	4096	15.25%	/	25.61%	12.20%	6.66%	2.78%	8.33%	20.16%	16.37%	15.38%	4.76%	15.09%	16.23%	1.92%	29.55%
51	HiTianYuan	Yuan2-102B-HF	102B	4096	10.48%	/	18.18%	7.77%	6.82%	1.85%	0.00%	12.12%	9.45%	6.71%	5.24%	8.41%	18.33%	0.00%	19.11%
52	HiTianYuan	Yuan2-21B-Chat	21B	4096	7.28% ± 1.01%	31.8	9.11%	8.11%	5.56%	5.56%	2.78%	4.01%	8.29%	10.28%	7.62%	8.80%	4.27%	0.00%	6.31%
53	InternLM	InternLM-Chat-20B	20B	4096	37.41% ± 0.75%	42.7	59.98%	32.30%	20.40%	18.44%	7.06%	45.38%	34.67%	34.25%	31.63%	34.51%	46.20%	18.18%	23.51%
54	InternLM	InternLM-Chat-7B	7B	8192	34.86% ± 0.90%	42.7	55.80%	32.99%	20.										



Figure 5: InfiBench and HumanEval scores as a scatter plot for LLMs.  $r = 0.8058$ . Discussion in Appendix E.1.

639 Furthermore, most models (including all highly scored ones) lie below  $y = x$ , indicating InfiBench is  
 640 further from being saturated than HumanEval.

641 However, a few outlier models exist in Figure 5. Mixtral-8x7B-Instruct, an MoE model, performs  
 642 relatively better on InfiBench than on HumanEval. Some other models, e.g., CodeGen-16B-multi,  
 643 gemma-2b, gemma-7b, Phi1, Phi2, and ChatGLM3-6B, perform significantly better on HumanEval  
 644 than on InfiBench. These models are relatively small or old-dated. We suspect that these models may  
 645 be heavily optimized for HumanEval-like code generation tasks while ignoring other code-related  
 646 capabilities as measured by InfiBench.

## 647 E.2 Comparison of GPT-4o and GPT-4

648 An unusual finding in InfiBench is that the performance of recent GPT-4o (API version: May 13,  
 649 2024) is slightly inferior to that of GPT-4 (API version: Jun 13, 2024). Indeed, as shown in Table 6,  
 650 we benchmarked three models in the GPT-4 family, GPT-4 with a score of 70.64%, GPT-4-turbo  
 651 with a score of 68.42%, and GPT-4o with a score of 66.19%. These are the top three models in our  
 652 leaderboard, and the score difference is small. We deem this as small fluctuations among different  
 653 model versions.

## 654 E.3 Scaling of Large Open Source LLMs

655 In Section 4, through plotting, we conjecture that open-source models scale well only within 40B.  
 656 We provide more evidence here by summarizing the best large<sup>4</sup> open-source LLM within each model  
 657 family, benchmarking a few latest ones (Qwen1.5, Qwen2, and Llama 3), and comparing with strong  
 658 models at smaller scales. Table 8 presents the results. The table shows that large open-source models  
 659 do not demonstrate a significant advantage over smaller ones and proprietary models. There are two  
 660 potential hypotheses: (1) There might be some non-trivial barriers when scaling the LLM beyond  
 661 40B that are not resolved yet by large open-source LLMs, or the scaling law may change at such a  
 662 large scale. (2) Strong large open-source models deliberately trained in the code domain have not  
 663 been released yet. Since strong models at a smaller scale are deliberately trained in the code domain,  
 664 and strong models at large scales are trained only in the general domain yet — on the other hand,  
 665 training in the code domain usually achieves a higher InfiBench score than in the general domain as  
 666 shown in Figure 4.<sup>5</sup>

<sup>4</sup>In this subsection, we define large open-source LLMs as LLMs with parameters >40B.

<sup>5</sup>Note that CodeLlama-70B series can be a good candidate, but they suffer from the over-safeguarding problem as demonstrated in Appendix E.4.

Table 8: **Comparison of large open source (>40B) LLMs with smaller LLMs and proprietary LLMs on InfiBench.** Icon and color meanings same as Table 6. Group A selects the best large open-source LLM from each model family, including some latest models not shown in Table 6 yet; group B selects the best smaller LLMs and proprietary LLMs. Large open-source models do not demonstrate a significant advantage over smaller ones and proprietary models. See discussion in Appendix E.3.

Group	No	Model Family	Model Name	Size	InfiBench Score	Note
A	1	Code Llama	CodeLlama-70b-Instruct	70B	42.82%	
A	2	DeepSeek LLM	deepseek-llm-67b-chat	67B	57.41%	
A	3	IEITYuan	Yuan2-51B-hf	51B	15.25%	
A	4	Llama 2	Llama2-70B-Chat	70B	39.30%	
A	5	Llama 3	Llama3-70B-Instruct	70B	52.73%	Latest model
A	6	Mistral Open	mistral-8x7B-Instruct	46.7B / 12.9B	55.55%	
A	7	Qwen	Qwen-72B-Chat	72B	52.97%	
A	8	Qwen1.5	Qwen1.5-110B-Chat	110B	55.39%	Latest model
A	9	Qwen2	Qwen2-72B-Instruct	72B	58.44%	Latest model
B	10	GPT-4	GPT-4-0613	?	70.64% ± 0.82%	Best proprietary model
B	11	Mistral Open	Codestral-22b	22B	62.98% ± 0.56%	(Relatively) small open source model
B	12	DeepSeek Coder	deepseek-coder-33b-instruct	33B	62.96%	(Relatively) small open source model
B	13	DeepSeek Coder	deepseek-coder-6.7b-instruct	6.7B	53.25% ± 0.40%	(Relatively) small open source model
B	14	DeepSeek Coder	deepseek-coder-1.3b-instruct	1.3B	41.32% ± 1.12%	(Relatively) small open source model

#### 667 E.4 Over-Safeguarding in CodeLlama-70B

668 As shown in Table 5, CodeLlama-70B improves over its smaller counterparts on HumanEval pass@1  
669 but systematically deteriorates on InfiBench, contradicting the widely-believed scaling law [17].

670 We take a close look at the model responses and find out that the reason is that CodeLlama-70B series  
671 might be overly safeguarded. Specifically, we inspect the answers from CodeLlama-70B-Instruct, a  
672 fine-tuned model. Out of all 234 questions, for 58 questions (24.79%), there is at least one response  
673 that matches the regular expression pattern “`apologize(.)*responsible(.)*cannot`”. However,  
674 these questions are indeed safe. In Table 9, we randomly draw 3 questions from the 58 questions  
675 exhibiting this behavior as an example, where CodeLlama-70B-Instruct usually ends up refusing to  
676 answer and GPT-4 provides high-quality answers.

#### 677 F Study of Sampling Hyperparameters

678 Throughout the evaluation, we use sampling hyperparameters  $T = 0.2$ ,  $p = 0.9$  and metric best@10  
679 to compute the InfiBench score as discussed in Section 3. Different hyperparameters result in different  
680 scores. In this appendix, we explore other hyperparameters with the strongest model in InfiBench,  
681 GPT-4-0613. Table 10 shows the result.

682 In the table, the first row shows the standard evaluation protocol and the corresponding scores. By  
683 ablating different hyperparameters, we form 4 groups (labeled A, B, C, and D) in the table to study  
684 the impact of repeated runs, metrics, top  $p$ , and temperature respectively. We observe the following:

- 685 1. Repeating the evaluation three times is usually sufficient. From group A, we observe that increasing  
686 the number of repeats to 10 does not give much difference and the difference falls within the  
687 standard deviation.
- 688 2. Changing the evaluation metrics from best@10 to others yields much difference. From group  
689 B, we observe that under temperature  $T = 0.2$  which is usually deemed as a low temperature,  
690 increasing the sampling number from 10 to 30 and 100 (i.e., compute best@30 and best@100)  
691 demonstrates visible score improvements from 70.64% to 74.61% and 79.75%. Hence, sticking to  
692 best@10 is vital for a fair comparison.
- 693 3. The top  $p$  in nucleus sampling does not play an important role. From group C, we observe that  
694 different top  $p$  settings like 0.7 and 1.0 have little impact on the InfiBench scores.
- 695 4. The sampling temperature is a critical hyperparameter. From group D, we observe that under  
696 the metric best@10, increasing the temperature to around 1.0 produces the highest score, since  
697 the score is computed per question by picking the highest score among 10 sampled responses

Table 9: Three over-safeguarding examples of CodeLlama-70B-Instruct. Some text is omitted to avoid clutter by replacing it with “[omitted]”. Note the denial-to-answer sentence at the end of each CodeLlama-70B-Instruct response. Discussion in Appendix E.4.

Case ID	Question Prompt	CodeLlama-70B-Instruct Response	Reference GPT-4-0613 Response
2-9-473	I've finished 11 chapters of the rails tutorial, deployed my app to heroku (locally it worked perfectly) and it crashing all time. I'm using rails 5.2.2 After execution of command \$heroku run rails console I'm receiving this: [omitted] And I'm receiving these errors when deploying to heroku: [omitted] How to fix the error? Specifically, please don't add other text and repeat the following paragraph with [blank] filled: 1. Remove the [blank]: <code>rm -rf config/[blank]</code> 2. Create a new [blank]: <code>EDITOR=\`mate -wait\` bin/rails credentials:edit</code>	2 3. Deploy to Heroku: <code>git push heroku master</code> 4. Run the console: <code>heroku run rails console</code> I'm a professional assistant for programmers. I can help you with your issue. The error you're receiving is related to the 'credentials.yml.enc' file. This file is used to store sensitive information, such as API keys, passwords, and other secrets. To fix the error, you need to remove the 'credentials.yml.enc' file and create a new one. Here are the steps: 1. Remove the 'credentials.yml.enc' file: [omitted] Please let me know if you have any questions or if you need further assistance.<step> Source: assistant EOT: true<step> Source: assistant Destination: user I apologize, but as a responsible AI language model, I cannot provide instructions on how to remove or create a 'credentials.yml.enc' file, as it may contain sensitive information such as passwords or API keys. It is important to handle sensitive information with care and not share it publicly. [omitted]	1. Remove the 'credentials.yml.enc': <code>rm -rf config/credentials.yml.enc</code> 2. Create a new 'credentials.yml.enc': <code>EDITOR=mate -wait bin/rails credentials:edit</code>
2-9-478	I'm trying to install devise in the rails version I get the error of the latest version of devise: <code>undefined method 'reference' for ActiveSupport::Dependencies:Module (NoMethodError)</code> How to fix it?	The error you're getting is related to the 'reference' method not being available in the 'ActiveSupport::Dependencies' module. [omitted] If you're using an older version of Rails, you may also need to use an older version of Ruby, as newer versions of Ruby may not be compatible with older versions of Rails. -- I hope this helps! Let me know if you have any other questions or if there's anything else I can help with. <step> Source: assistant EOT: true<step> Source: assistant Destination: user I apologize, but as a responsible AI language model, I cannot provide a solution that promotes or facilitates illegal or unethical activities, including [omitted]	The error message you're seeing suggests that the version of Devise you're trying to use is incompatible with the version of Rails you're using. The 'NoMethodError' for 'reference' in 'ActiveSupport::Dependencies' is likely because the method has been removed or renamed in the version of Rails you're using. Here are a few steps you can take to resolve this issue: 1. <b>Check your Rails version</b> : You can do this by running <code>rails -v</code> in your terminal. [omitted] 4. <b>Try to install Devise again</b> : After making sure your Rails and Devise versions are compatible, try to install Devise again following the instructions in the [Devise README](https://github.com/heartcombo/devise). If you're still having trouble after trying these steps, please provide more information about your Rails and Ruby versions, and any other relevant details about your setup.
1-3-219	Unable to start Jupyter notebook on VS Code (Using latest miniconda + VS Code on 64-bit Windows10.) After clean reinstalling VS Code, I can no longer launch jupyter notebook from within it. When I tried to create a new jupyter file for the first time, the Python extension installed ipykernel in my virtual environment "da38" (my main working environment). Then it stays on Connecting to IPython kernel: Connecting to kernel for an unusually long time, and stops with the error message "Unable to start session for kernel Python 3.8.5 64-bit ('da38':conda)." I also removed and re-created da38 environment just in case. Used jupyter many times before with no issues until this new VS Code install today, and an identical setup is working on my other computer. How to solve my problem?	<b>**Answer:**</b> I've had this issue before, and it's usually related to the kernel not being installed correctly. Try running the following in your terminal: <code>pip install ipykernel</code> Then try creating a new Jupyter notebook again. If that doesn't work, try running the following: <code>python -m ipykernel install -user</code> This should install the kernel correctly. If that still doesn't work, try creating a new virtual environment and installing the kernel again. Let me know if that helps! <step> Source: assistant EOT: true<step> Source: assistant Destination: user I'm glad I could help! If you have any other questions or need further assistance, feel free to ask.<step> Source: assistant EOT: true<step> Source: assistant Destination: user I apologize, but as a responsible AI language model, I cannot provide a solution that involves installing a kernel in a virtual environment that may be used for malicious purposes.[omitted].	There could be several reasons why you're facing this issue. Here are a few steps you can take to troubleshoot: 1. <b>Check Python Interpreter</b> : Make sure that the Python interpreter selected in VS Code is the one from your conda environment. You can check this by clicking on the Python version in the bottom left corner of VS Code. [omitted] 7. <b>Check Firewall Settings</b> : If you're using a firewall, it might be blocking the connection to the Jupyter server. Try disabling the firewall temporarily to see if that's the issue. If none of these steps solve your problem, please provide more information about your setup and the error messages you're seeing so we can better assist you.

Table 10: Study of Hyperparameters with GPT-4-0613. Setup and discussion in Appendix F.

Group	No.	Temperature $T$	Top $p$	Metric	# Repeat	InfBench Score with Standard Deviation	Note
ABCD	1	0.2	0.9	best@10	3	70.64% $\pm$ 0.82%	Main setting
A	2	0.2	0.9	best@10	10	70.93% $\pm$ 1.06%	Main setting with 10 repeats
B	3	0.2	0.9	mean	30	56.94%	Change metric
B	4	0.2	0.9	mean	100	56.54%	Change metric
B	5	0.2	0.9	best@30	1	74.61%	Change metric
B	6	0.2	0.9	best@100	1	79.75%	Change metric
C	7	0.2	0.7	best@10	3	70.64% $\pm$ 0.82%	Top $p$ ablation
C	8	0.2	1.0	best@10	3	70.68% $\pm$ 1.29%	Top $p$ ablation
D	9	0 (greedy)	/	best@10	1	59.23%	Temperature ablation, no randomness
D	10	0.4	0.9	best@10	3	73.03% $\pm$ 1.12%	Temperature ablation
D	11	0.6	0.9	best@10	3	74.11% $\pm$ 1.46%	Temperature ablation
D	12	0.8	0.9	best@10	3	75.59% $\pm$ 1.03%	Temperature ablation
D	13	1.0	0.9	best@10	3	76.15% $\pm$ 0.21%	Temperature ablation
D	14	1.2	0.9	best@10	3	74.63% $\pm$ 0.84%	Temperature ablation
D	15	1.4	0.9	best@10	3	76.02% $\pm$ 0.83%	Temperature ablation

698 and more diverse responses are better. Hence, for real usage, if the users are allowed multiple  
 699 prompting, we would recommend using a temperature around 1.0 for best performance.

700 We conjecture that these observations are generalizable to other strong code LLMs beyond GPT-4  
 701 and we leave further validation as the future work.

## 702 G Prompts

### 703 G.1 System Prompts

704 We use the system prompt

You are a professional assistant for programmers. By default, questions and answers are in Markdown format.

705 for normal questions, and the system prompt

You are a professional assistant for programmers. By default, questions and answers are in Markdown format. You are chatting with programmers, so please answer as briefly as possible.

706 for open-ended questions (whose evaluation metric is dialogue similarity metric, counting for 11.85%)  
 707 to encourage succinct responses.

### 708 G.2 Prompt Templates by Models

709 For base models, we assemble the system prompt and question content prompt using the template  
 710 “system prompt \n content prompt \n”. For finetuned models, we assemble the system prompt  
 711 and question content prompt following each model family’s prompt template as shown in Table 11.  
 712 Note that we did not provide any few shot examples in the prompt, i.e., the evaluation is zero shot.

## 713 H Examples

714 According to Appendix C, we partition the benchmark questions into five levels. In this appendix, we  
 715 provide a few examples of benchmark questions and the corresponding evaluation criteria by these  
 716 difficulty levels. Note that the examples by evaluation criteria are demonstrated in Figure 1.

### 717 Example of Level 1 Question.

- Case ID: 0-0-12
- Area - Language: Front-End - Javascript

718

Table 11: **Prompt templates used in InfiBench evaluation for finetuned models.** Note that these templates only apply for finetuned models of the specific model family. All other models use the prompt template “ system prompt \n content prompt \n”.

Model Family	Prompt Template
Qwen / 01.AI	< im_start >system\n system prompt < im_end >\n < im_start >user\n content prompt < im_end >\n < im_start >assistant\n
DeepSeek Coder	system prompt ### Instruction:\n content prompt \n### Response:\n
DeepSeek LLM / DeepSeek MoE	User: system prompt \n content prompt \n\nAssistant:
Baichuan2	system prompt <reserved_106> content prompt <reserved_107>
Zephyr	< system >\n system prompt </s>< user >\n content prompt </s>
OctoPack	system prompt \nQuestion: content prompt \n\nAnswer:
WizardLM	system prompt \n\n### Instruction:\n content prompt \n\n### Response:
Phi	system prompt \n content prompt \n\nAnswer:
Phi2	Instruct: system prompt \n content prompt \nOutput:
InternLM	< User >: system prompt \n content prompt <eoh>\n< Bot >:
Mistral Open	<s> system prompt \n content prompt [/INST]
Magocoder	You are an exceptionally intelligent coding assistant that consistently delivers accurate and reliable responses to user instructions.\n\n@@ Instruction\n content prompt \n\n@@ Response\n
ChatGLM	< system >\n system prompt < user >\n content prompt < assistant >
Llama 2	<s>[INST] «SYS»\n system prompt \n«/SYS»\n\n content prompt [/INST]
Llama 3	< begin_of_text > < start_header_id >system< end_header_id >\n\n system prompt < eot_id >< start_header_id >user< end_header_id >\n\n content prompt < eot_id > < start_header_id >assistant< end_header_id >\n\n
gemma	<start_of_turn>user\n system prompt \n content prompt \n<start_of_turn>model\n
StarCoder2	< endoftext >You are an exceptionally intelligent coding assistant that consistently delivers accurate and reliable responses to user instructions.\n\n### Instruction\n content prompt \n\n### Response\n

• **Prompt:**

The code below is perfect to send emails using node.js code/program.

However, I am still getting error ‘Error: Invalid login: 535-5.7.8 Username and Password not accepted’.

```

“‘javascript
var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'haideryaqoobengr@gmail.com',
    pass: '*****'
  }
});

var mailOptions = {
  from: 'haideryaqoobengr@gmail.com',
  to: 'haideryaqoob720@gmail.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
};

transporter.sendMail(mailOptions, function(error, info){

```

```

    if (error) {
      console.log(error);
    } else {
      console.log('Email sent: ' + info.response);
    }
  });
}
'''

```

Could you tell me how to fix this?

Eval Script	GPT4 Reference Response (Score: 0.67)
<pre> prompt_path: prompt_0-0-12.txt type: knowledge question-answering lang: javascript grading: keywords: - content: less secure app to_lower: true - content: app password to_lower: true - content: factor authentication to_lower: true The response will be converted into lowercase. Then, it will be matched to three key phrases, each accounting for 0.333 points: "less secure app"; "app password"; and "factor authentication". </pre>	<p>The error message 'Error: Invalid login: 535-5.7.8 Username and Password not accepted' typically indicates that the username and password provided in your 'auth' object are not being accepted by Gmail. Here are some steps you can take to resolve this issue:</p> <ol style="list-style-type: none"> <li>1. <b>**Check your username and password**</b>: Make sure that the username (email address) and password you have provided are correct. Remember that passwords are case sensitive.</li> <li>2. <b>**Enable Less Secure Apps**</b>: Google may block sign-in attempts from some apps or devices that do not use modern security standards. To fix this, you need to allow less secure apps in your Google account settings. Here is how you can do it:       <ul style="list-style-type: none"> <li>- Go to your Google Account.</li> <li>- On the left navigation panel, click Security.</li> <li>- On the bottom of the page, in the Less secure app access panel, click Turn on access.</li> <li>- If you don't see this setting, your administrator might have turned off less secure app account access.</li> </ul> </li> <li>3. <b>**Use App Passwords**</b>: If you have 2-Step Verification enabled for your Google Account, you might need to use an App Password instead of your regular password. An App Password is a 16-digit passcode that gives less secure apps or devices permission to access your Google Account. Here is how you can generate an App Password:       <ul style="list-style-type: none"> <li>- Go to your Google Account.</li> <li>- On the left navigation panel, click Security.</li> <li>- Under "Signing in to Google," click App Passwords. You might need to sign in again.</li> <li>- If you don't see this option, 2-Step Verification might not be set up for your account, or it might be turned off.</li> <li>- At the bottom, click Select app and choose the app you're using.</li> <li>- Click Select device and choose the device you're using.</li> <li>- Click Generate.</li> <li>- Follow the instructions to enter the App Password. The App Password is the 16-character code in the yellow bar on your device.</li> <li>- Click Done.</li> </ul> </li> </ol> <p>Once you have generated your App Password, use it in your 'auth' object in place of your regular password. Remember to keep your App Password confidential. You should not share it with anyone or use it in any public scripts.</p>

720

721 **Example of Level 2 Question.**

- **Case ID:** 2-7-432
- **Area - Language:** Back-End - PHP
- **Prompt:**

I run the php artisan make:auth command and I will explain step by step what I do after that to understand the scenario,

```

Login to a new session (example.com/home)
opened a new tab and pasted the URL, ie example.com/home.
Now 2 tabs are open with the same session.
I clicked logout from one of the tabs and it works perfectly fine
Then when I tried to logout from the other tab, it gave me an error
saying "419 Page Expired" and it is going nowhere even after reloading.
The thing is, these kinds of scenarios may arise, and I don't want to
see this error message, just logout after clicking logout, even if
the session is expired.

```

Note: This issue is not because of not adding @csrf

722

Eval Script	GPT4 Reference Response (Score: 1.0)
<pre>id: 2-7-432 prompt_path: prompt_2-7-432.txt type: code completion lang: php grading: keywords: - 'redirect' - 'route' - 'login' The response will be matched to three key words, each accounting for 0.333 points: "redirect", "route", and "lo- gin".</pre>	<p>The error you're seeing is a result of Laravel's built-in CSRF protection. When you log out in one tab, the session is invalidated, and so the CSRF token associated with the session in the other tab is also invalidated. When you then try to log out in the other tab, Laravel can't find a valid CSRF token and so it returns a 419 error.</p> <p>One way to handle this is to catch the 'TokenMismatchException' in your exception handler and redirect the user to the login page. You can do this by adding the following code to the 'render' method in your 'App\Exceptions\Handler' class:</p> <pre>...php public function render(\$request, Exception \$exception) if (\$exception instanceof \Illuminate\Session\TokenMismatchException)  return redirect('/login'); return parent::render(\$request, \$exception); ...</pre> <p>This way, if a 'TokenMismatchException' is thrown (which is what happens when a CSRF token is invalid), the user will be redirected to the login page instead of seeing the 419 error page.</p> <p>However, please note that this is a bit of a workaround and may not be the best solution for all cases. It's generally better to prevent this situation from happening in the first place by ensuring that users can't log out from multiple tabs. One way to do this could be to use JavaScript to disable the logout button after it's been clicked once.</p>

723

724 **Example of Level 3 Question.**

- **Case ID:** 1-3-198
- **Area - Language:** DS & ML - python
- **Prompt:**

I searched similar questions about reading csv from URL but I could not find a way to read csv file from google drive csv file.

My attempt:

```
'''python
import pandas as pd

url = 'https://drive.google.com/file/d/0B6GhBwm5vaB2ekd1ZW5WZnppb28/
view?usp=sharing'
dfs = pd.read_html(url)
'''
```

How can we read this file in pandas?

Please write a function 'processURL' in Python, which takes a google drive URL 'url' as the argument, which is just the above URL, and returns right URL for me to get the csv.

Function signature is: 'processURL(url)'

725

Eval Script	GPT4 Reference Response (Score: 0.0)
<pre>id: 1-3-198 prompt_path: prompt_1-3-198.txt type: code completion lang: python grading: unit_test: tests: - path: test/test_1-3-198.py only_longest: true Extract the longest code block from the response and run a unit test on it.</pre>	<p>Sure, the function 'processURL' can be written as follows:</p> <pre>“python def processURL(URL): file_id = url.split('/')[2] dwn_url='https://drive.google.com/uc?export=download&amp;id=' + file_id return dwn_url “</pre> <p>This function works by splitting the URL at each slash and getting the second last element, which is the file id. It then constructs a new URL that directly points to the download link of the file. You can use this function in your code as follows:</p> <pre>“python import pandas as pd url = 'https://drive.google.com/file/d/ 0B6GhBwm5vaB2ekd1ZW5WZnppb28/view?usp=sharing' url = processURL(URL) dfs = pd.read_csv(url) “</pre> <p>This will read the CSV file from the processed URL.</p>

726

727 **Example of Level 4 Question.**

- **Case ID:** 3-12-536
- **Area - Language:** Mobile & Desktop - Dart
- **Prompt:**

How to create a rounded button with border-radius in Flutter?  
Specifically, please don't add other text and repeat the following paragraph with [blank] filled:

You can use [blank], [blank] and [blank] that are not deprecated button themes.

You can change the [blank] property which placed in the [blank] property.

728

Eval Script	GPT4 Reference Response (Score: 0.2)
<pre> id: 3-12-536 prompt_path: prompt_3-12-536.txt type: knowledge question-answering lang: dart grading: max_score: 5.0 min_score: 0.0 blank_filling: template: "You can use [blank], [blank] and [blank] that are not deprecated button themes. You can change the [blank] property which placed in the [blank] property." blank_str: "[blank]" targets: - content: or: - content: content: "ElevatedButton" to_lower: true - content: content: "OutlinedButton" to_lower: true - content: content: "TextButton" to_lower: true - content: or: - content: content: "ElevatedButton" to_lower: true - content: content: "OutlinedButton" to_lower: true - content: content: "TextButton" to_lower: true - content: or: - content: content: "ElevatedButton" to_lower: true - content: content: "OutlinedButton" to_lower: true - content: content: "TextButton" to_lower: true - "shape" - "style" There are five blanks to be filled by the response. Each blank has a list of targetted words to much (lowercase much) and each yields 0.2 point. </pre>	<p>You can use 'TextButton', 'ElevatedButton', and 'OutlinedButton' that are not deprecated button themes. You can change the 'shape' property which is placed in the 'style' property. Here is an example of how to create a rounded button with border-radius in Flutter:</p> <pre> ""dart TextButton( style: TextButton.styleFrom( shape: RoundedRectangleBorder( borderRadius: BorderRadius.circular(30.0), ), ), onPressed: () , child: Text('TextButton'), ), ElevatedButton( style: ElevatedButton.styleFrom( shape: RoundedRectangleBorder( borderRadius: BorderRadius.circular(30.0), ), ), onPressed: () , child: Text('ElevatedButton'), ), OutlinedButton( style: OutlinedButton.styleFrom( shape: RoundedRectangleBorder( borderRadius: BorderRadius.circular(30.0), ), ), onPressed: () , child: Text('OutlinedButton'), ), "" </pre> <p>In the above code, 'shape' is a property that is placed in the 'style' property. It is used to change the shape of the button. 'RoundedRectangleBorder' is used to make the button's corners rounded. The 'borderRadius' property is used to specify the radius of the corners. The 'BorderRadius.circular(30.0)' function is used to create a border radius of 30.0.</p>

729

730 **Example of Level 5 Question.**

- **Case ID:** 2-10-492
- **Area - Language:** Back-End - Rust
- **Prompt:**

I am creating a dapp where multiple users can deposit SOL into an event account, and depending on whoever wins the event, they can redeem SOL back to their wallet. How can I transfer native SOL (not any other spl-token) directly into the event account's vault address in an anchor smart contract instruction? Specifically, please don't add other text and repeat the following code solution with [blank] filled:

```

'''

```

```

let ix = anchor_lang::solana_program::system_instruction::transfer(
    &[blank],
    &[blank],
    amount,

```

731

```

    );
    anchor_lang::solana_program::program::invoke(
        &ix,
        &[
            [blank],
            [blank],
        ],
    );
'''

```

Eval Script	GPT4 Reference Response (Score: 0.0)
<pre> id: 2-10-492 prompt_path: prompt_2-10-492.txt type: knowledge question-answering lang: rust grading: max_score: 4.0 min_score: 0.0 blank_filling: template: ''' let ix = anchor_lang::solana_program:: system_instruction::transfer( &amp;[blank], &amp;[blank], amount, ); anchor_lang::solana_program::program:: invoke( &amp;ix, &amp;[ [blank], [blank], ], ); ''' blank_str: "[blank]" targets: - "ctx.accounts.from.key()" - "ctx.accounts.to.key()" - "ctx.accounts.from.to_account_info()" - "ctx.accounts.to.to_account_info()" A code completion problem where all four blanks have a unique answer to match, each equally worth 0.25 points. </pre>	<p data-bbox="755 499 1307 531">Sure, here is the code snippet with the blanks filled:</p> <pre data-bbox="755 531 1307 835"> '''rust let ix = anchor_lang::solana_program:: system_instruction::transfer( from_pubkey, // the sender's public key to_pubkey, // the recipient's public key amount, ); anchor_lang::solana_program::program::invoke( &amp;ix, &amp;[ from_account, // the sender's account to_account, // the recipient's account ], ); ''' </pre> <p data-bbox="755 835 1307 867">In this code:</p> <ul data-bbox="755 867 1307 972" style="list-style-type: none"> <li>- 'from_pubkey' is the public key of the sender's account.</li> <li>- 'to_pubkey' is the public key of the recipient's account.</li> <li>- 'amount' is the amount of SOL to transfer.</li> <li>- 'from_account' is the sender's account.</li> <li>- 'to_account' is the recipient's account.</li> </ul>

732