LiveBench: A Challenging, Contamination-Free LLM Benchmark

Colin White* [1], Samuel Dooley*[1], Manley Roberts*[1], Arka Pal*[1], Benjamin Feuer[2], Siddhartha Jain[3], Ravid Shwartz-Ziv[2], Neel Jain[4], Khalid Saifullah[4], Siddartha Naidu[1], Chinmay Hegde[2], Yann LeCun[2], Tom Goldstein[4], Willie Neiswanger[5], Micah Goldblum[2]
[1] Abacus.AI, [2] NYU, [3] Nvidia, [4] UMD, [5] USC

Abstract

Test set contamination, wherein test data from a benchmark ends up in a newer model's training set, is a well-documented obstacle for fair LLM evaluation and can quickly render benchmarks obsolete. To mitigate this, many recent benchmarks crowdsource new prompts and evaluations from human or LLM judges; however, these can introduce significant biases, and break down when scoring hard questions. In this work, we introduce a new benchmark for LLMs designed to be immune to both test set contamination and the pitfalls of LLM judging and human crowdsourcing. We release `LiveBench`, the first benchmark that (1) contains frequently-updated questions from recent information sources, (2) scores answers automatically according to objective ground-truth values, and (3) contains a wide variety of challenging tasks, spanning math, coding, reasoning, language, instruction following, and data analysis. To achieve this, `LiveBench` contains questions that are based on recently-released math competitions, arXiv papers, news articles, and datasets, and it contains harder, contamination-free versions of tasks from previous benchmarks such as Big-Bench Hard, AMPS, bAbI, and IFEval. We evaluate many prominent closed-source models, as well as dozens of open-source models ranging from 0.5B to 110B in size. LiveBench is difficult, with top models achieving below 60% accuracy. We release all questions, code, and model answers. Questions will be added and updated on a monthly basis, and we will release new tasks and harder versions of tasks over time so that LiveBench can distinguish between the capabilities of LLMs as they improve in the future. We welcome community engagement and collaboration for expanding the benchmark tasks and models.

1 Introduction

In recent years, as large language models (LLMs) have risen in prominence, it has become increasingly clear that traditional machine learning benchmark frameworks are no longer sufficient to evaluate new models. Benchmarks are typically published on the internet, and most modern LLMs include large swaths of the internet in their training data. If the LLM has seen the questions of a benchmark during training, its performance on that benchmark will be artificially inflated [14, 16, 23, 46], hence making many LLM benchmarks unreliable. Recent evidence of test set contamination includes the observation that LLMs' performance on Codeforces plummet after the training cutoff date of the LLM [27, 46], and before the cutoff date, performance is highly correlated with the number of times

*Correspondence to: `colin@abacus.ai`, `samuel@abacus.ai`, `goldblum@nyu.edu`. Sponsored by Abacus.AI.

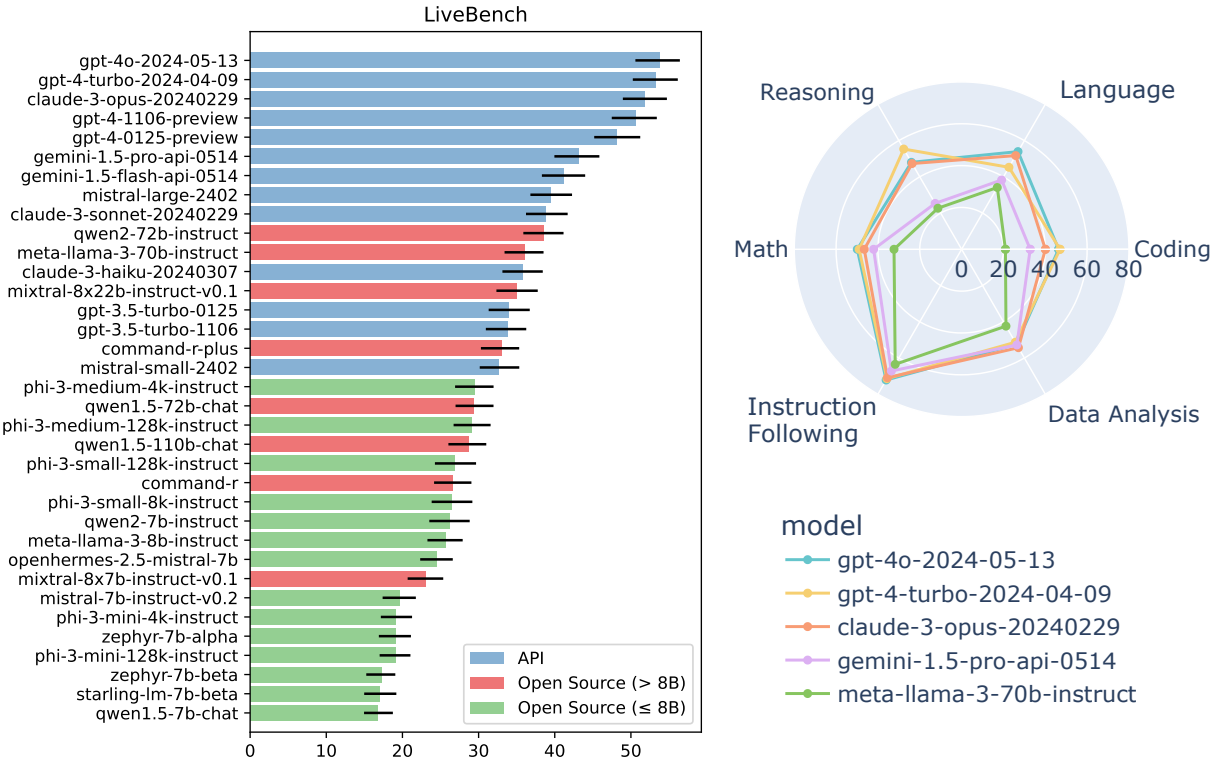

Figure 1: Results on `LiveBench` for all models, showing 95% bootstrap confidence intervals (left). A radar plot for select models across `LiveBench`'s six categories demonstrating the that ordering of top models varies between each category (right).

the problem appears on GitHub [46]. Similarly, a recent hand-crafted variant of the established math dataset, GSM8K, shows evidence that several models have overfit to this benchmark [12, 64].

To lessen dataset contamination, benchmarks using LLM or human prompting and judging have become increasingly popular [10, 27, 34, 65]. However, using these techniques comes with significant downsides. While LLM judges have multiple advantages, such as their speed and ability to evaluate open-ended questions, they are prone to making mistakes and can have several biases. For example, we will show in Section 3 that for challenging reasoning and math problems, the pass/fail judgments from `GPT-4-Turbo` **have an error rate of up to 46%**. Furthermore, LLMs often favor their own answers over other LLMs, and LLMs favor more verbose answers [17, 34, 35]. Additionally, using humans to provide evaluations of LLMs can inject biases such as formatting of the output, and the tone and formality of the writing [10]. Using humans to generate questions also presents limitations. Human participants might not ask diverse questions, may favor certain topics that do not probe a model's general capabilities, or may construct their prompts poorly [65].

In this work, we introduce a framework for benchmarking LLMs designed to be immune to both test set contamination and the pitfalls of LLM judging and human crowdsourcing. We use this framework to create `LiveBench`, the first benchmark with these three desiderata: *(1)* `LiveBench` contains frequently-updated questions based on recent information sources; *(2)* `LiveBench` is scored automatically according to the objective ground-truth without the use of an LLM judge; and *(3)* `LiveBench` questions are drawn from a diverse set of six categories. We ensure *(2)* by only including

questions that have an objectively correct answer in LiveBench. LiveBench questions are *difficult*; no current model achieves higher than 60% accuracy. Questions will be added and updated on a monthly basis, and we will release new tasks and harder versions of tasks over time so that LiveBench can distinguish between the capabilities of LLMs as they improve in the future.

Overview of tasks. LiveBench currently consists of 18 tasks across 6 categories: math, coding, reasoning, language, instruction following, and data analysis. Each task falls into one of two types: *(1)* tasks which use an information source for their questions, e.g., data analysis questions based on recent Kaggle datasets, or fixing typos in recent arXiv abstracts; and *(2)* tasks which are more challenging or diverse versions of existing benchmark tasks, e.g., from Big-Bench Hard [50], IFEval [66], bAbI [59], or AMPS [25]. The categories and tasks included in LiveBench are:

- **Math:** questions from high school math competitions from the past 12 months (AMC12, AIME, USAMO, IMO, SMC), as well as harder versions of AMPS questions [26]
- **Coding:** code generation questions from Leetcode and AtCoder (via LiveCodeBench [27]), as well as a novel code completion task
- **Reasoning:** a harder version of Web of Lies from Big-Bench Hard [50], a harder version of PathFinding from bAbI [59], and Zebra Puzzles (e.g., [28])
- **Language Comprehension:** three tasks: Connections word puzzles, a typo-fixing task, and a movie synopsis unscrambling task from recent movies on IMDb and Wikipedia
- **Instruction Following:** four tasks to paraphrase, simplify, summarize, or generate stories about recent new articles from The Guardian [24], subject to one or more instructions such as word limits or incorporating specific elements in the response
- **Data Analysis:** three tasks using recent datasets from Kaggle and Socrata: table reformatting (among JSON, JSONL, Markdown, CSV, TSV, and HTML), predicting which columns can be used to join two tables, and predicting the correct type annotation of a data column

We evaluate dozens of models, including proprietary models as well as open-source models with sizes ranging from 0.5B to 8x22B. We release all questions, code, and model answers, and we welcome community engagement and collaboration. Our codebase is available at https://github.com/livebench/livebench, and our leaderboard is available at https://livebench.ai.

2 LiveBench Description

In this section, we introduce LiveBench. It currently has six categories: math, coding, reasoning, data analysis, instruction following, and language comprehension. Categories are diverse with two to four tasks per problem. Each task either includes recent information sources (such as very recent news articles, movie synopses, or datasets) or is a more challenging, more diverse version of an existing benchmark task.

Each task is designed to span a range of difficulty, from easy to very challenging, while loosely aiming for a 30-70% success rate on the top models. Prompts are tailored for each category and task but typically include the following: zero-shot chain of thought [30, 58], asking the model to make its best guess if it does not know the answer, and asking the LLM to output its final answer in a way that is easy to parse, such as in **double asterisks**. In the following sections, we give an overview description of each task from each category.

2.1 Math Category

Evaluating the mathematical abilities of LLMs has been one of the cornerstones of recent research in LLMs, featuring prominently in many releases and reports [6, 7, 41, 45]. Our benchmark includes math questions of three types: questions from recent high school math competitions, fill-in-the-blank questions from recent proof-based USAMO and IMO problems, and questions from our new, harder version of the AMPS dataset [26].

Our first two math tasks, `Competitions` and `Proof Competition`, use expert human-designed math problems that offer a wide variety in terms of problem type and solution technique. First, we include questions from AMC12 2023, SMC 2023, and AIME 2024 in `Competitions` and from USAMO 2023 and IMO 2023 in `Proof Competitions`. These are challenging and prestigious competitions for high school students in the USA (AMC, AIME, USAMO), in the UK (SMC), and internationally (IMO). The competitions test mathematical problem solving with arithmetic, algebra, counting, geometry, number theory, probability, and other secondary school math topics [19].

Finally, we release synthetically generated math questions in the `AMPS_Hard` task. This task is inspired by the math question generation used to create the MATH and AMPS datasets [26]. We generate harder questions by drawing random primitives, using a larger and more challenging distribution than AMPS across the 10 hardest tasks within AMPS.

2.2 Coding Category

The coding ability of LLMs is one of the most widely studied and sought-after skills for LLMs [27, 33, 40]. We include two coding tasks in `LiveBench`: a modified version of the code generation task from LiveCodeBench (LCB) [27], and a novel code completion task combining LCB problems with partial solutions collected from GitHub sources.

In the `LCB Generation` task, we assess a model's ability to parse a competition coding question statement and write a correct answer. We include 50 questions from LiveCodeBench [27] which has several tasks to assess the coding capabilities of large language models.

The `Completion` task specifically focuses on the ability of models to complete a partially correct solution—assessing whether a model can parse the question, identify the function of the existing code, and determine how to complete it. We use LeetCode medium and hard problems from LiveCodeBench's [27] April 2024 release, combined with matching solutions from https://github.com/kamyu104/LeetCode-Solutions, omitting the last 15% of each solution and asking the LLM to complete the solution.

2.3 Reasoning Category

The reasoning ability of large language models is another highly-benchmarked and analyzed skill of LLMs [50, 58, 61]. In `LiveBench`, we include three reasoning tasks: our harder versions of tasks from Big-Bench Hard [50] and bAbI [59], and Zebra puzzles.

The `Web of Lies v2` task is an advancement of the similarly named task included in Big-Bench [5] and Big-Bench Hard [50]. The task is to evaluate the truth value of a random Boolean function expressed as a natural-language word problem. Already by October 2022, LLMs achieved near 100% on this task, and furthermore, there are concerns that Big-Bench tasks leaked into the training data of LLMs such as GPT-4, despite using canary strings [41]. For `LiveBench`, we create a new, significantly harder version by including additional deductive components and red herrings.

Next, we include a harder version of the `PathFinding` task from bAbI [59] that we call 'House Traversal'. The original task consists of sentences of the form, 'The bedroom is West of the kitchen. The kitchen is South of the garden', asking the model where a room is in relation to another room. We make this task significantly harder by adding a distinct person in each room and asking the LLM who a person would see if they traverse in a few directions.

The final reasoning task we include is `Zebra Puzzles`, a well-known reasoning task [28] that tests the ability of the model to follow a set of statements that set up constraints, and then logically deduce the requested information. We build on an existing repository for procedural generation of Zebra puzzles [42]; the repository allows for randomizing the number of people, the number of attributes, and the set of constraint statements provided. Below, we provide an example question from the `Zebra Puzzles` task.

An example question from the `Zebra Puzzle` task.

There are 3 people standing in a line numbered 1 through 3 in a left to right order.
Each person has a set of attributes: Food, Nationality, Hobby.
The attributes have the following possible values:
- Food: nectarine, garlic, cucumber
- Nationality: chinese, japanese, thai
- Hobby: magic-tricks, filmmaking, puzzles
and exactly one person in the line has a given value for an attribute.
Given the following premises about the line of people:
- the person that likes garlic is on the far left
- the person who is thai is somewhere to the right of the person who likes magic-tricks
- the person who is chinese is somewhere between the person that likes cucumber and the person who likes puzzles
Answer the following question: What is the hobby of the person who is thai? Return your answer as a single word, in the following format: **X**, where X is the answer.

2.4 Data Analysis Category

While LLMs' abilities in the previous three categories have been widely studied, we also include a category that is significantly less studied but is a practical application of LLMs that requires more attention: data analysis tasks. We include three tasks in which the LLM assists in data analysis or data science: column type annotation, table join prediction, and table reformatting. Each question makes use of a recent dataset from Kaggle or Socrata.

The first task is to predict the type of a column of a data table. To create a question for the column table annotation task (`CTA`), we randomly sample a table and randomly sample a column from that table. We use the actual column name of that column as the ground truth and then retrieve some column samples from that column. We provide the name of all the columns from that table and ask the LLM to select the true column name from those options.

Data analysts often also require a table to be reformatted from one type to another, e.g., json to CSV or XML to TSV. We emulate that task in `TableReformat` by providing a table in one format and asking the LLM to reformat it into the target format.

Finally, another common application of LLMs in data analysis is to perform table joins. In the `TableJoin` task, each question prompts an LLM to decide which columns can be used to join two different CSV tables.

2.5 Instruction Following Category

An important ability of an LLM is its capability to follow instructions. To this end, we include instruction-following questions in our benchmark, inspired by IFEval [66], which is an instruction-following evaluation for LLMs containing verifiable instructions such as "write more than 300 words" or "Finish your response with this exact phrase: {end_phrase}." While IFEval used a list of 25 verifiable instructions, we use a subset of 16 that excludes instructions that do not reflect real-world use-cases. See Appendix Table 4. Furthermore, in contrast to IFEval, which presents only the task and instructions with a simple prompt like "write a travel blog about Japan", we provide the models with an article from The Guardian [24], asking the models to adhere to multiple randomly-drawn instructions while asking the model to complete one of four tasks related to the article: `Paraphrase`, `Simplify`, `Story Generation`, and `Summarize`. We score tasks purely by their adherence to the instructions.

2.6 Language Comprehension Category

Finally, we include multiple language comprehension tasks. These tasks assess the language model's ability to reason about language itself by, (1) completing word puzzles, (2) fixing misspellings but leaving other stylistic changes in place, and (3) reordering scrambled plots of unknown movies.

First, we include the `Connections` category. Connections is a word puzzle popularized by the New York Times (although similar ideas have existed previously). In this task, we present questions of varying levels of difficulty with 8, 12, and 16-word varieties. The objective of the game is to sort the words into sets of four words, such that each set has a 'connection' between them, e.g., types of fruits, homophones, or words that come after the word 'fire'. Due to the variety of possible connection types, this task is challenging for LLMs, as shown by prior work that tested the task on the GPT family of models [55].

Next, we include the `Typos` task. The idea behind this task is inspired by the common use case for LLMs in which a user asks the LLM to identify typos and misspellings in some written text but to leave other aspects of the text unchanged. It is common for the LLM to impose its own writing style onto that of the input text, such as switching from British to US spellings or adding the serial comma, which may not be desirable. We create the questions for this task from recent ArXiv abstracts, which we ensure originally have no typos, by programmatically injecting common human typos into the text. Below is an example question from the `Typos` task.

An example question from the `Typos` task.

Please output this exact text, with no changes at all except for fixing the misspellings. Please leave all other stylistic decisions like commas and US vs British spellings as in the original text.

We inctroduce a Bayesian estimation approach forther passive localization of an accoustic source in shallow water using a single mobile receiver. The proposed probablistic focalization method estimates the timne-varying source location inther presense of measurement-origin uncertainty. In particular, probabilistic data assocation is performed to match tiome-differences-of-arival (TDOA) observations extracted from the acoustic signal to TDOA predicitons provded by the statistical modle. The performence of our approach is evaluated useing rela acoustic data recorded by a single mobile reciever.

Table 1: **LiveBench results across the 16 top-performing models.** We display in this table the highest-performing models on `LiveBench`, outputting the results on each main category, as well as each model's overall performance. See Table 3 for the results on all 43 models.

Model	LiveBench Score	Coding	Data Analysis	Instruction Following	Language	Math	Reasoning
gpt-4o-2024-05-13	**53.8**	46.4	52.4	**72.2**	**53.9**	**49.9**	48.0
gpt-4-turbo-2024-04-09	53.2	**47.1**	51.3	71.4	45.3	49.0	**55.3**
claude-3-opus-20240229	51.8	40.1	**54.3**	70.9	51.7	46.5	47.3
gpt-4-1106-preview	50.6	44.4	51.3	69.4	48.4	47.6	42.7
gpt-4-0125-preview	48.2	44.1	54.1	63.9	43.6	42.7	40.7
gemini-1.5-pro-api-0514	43.1	32.8	52.8	67.2	38.3	42.1	25.3
gemini-1.5-flash-api-0514	41.1	39.1	44.0	63.0	30.7	38.5	31.3
mistral-large-2402	39.4	26.8	42.6	68.2	28.7	32.2	38.0
claude-3-sonnet-20240229	38.9	25.2	44.6	65.0	38.1	29.6	30.7
qwen2-72b-instruct	38.5	31.8	26.2	68.3	29.2	43.4	32.0
meta-llama-3-70b-instruct	36.0	20.9	42.4	63.5	34.1	32.3	22.7
claude-3-haiku-20240307	35.8	24.5	41.5	64.0	30.1	25.7	28.7
mixtral-8x22b-instruct-v0.1	35.0	33.1	30.3	63.2	26.5	26.9	30.0
gpt-3.5-turbo-0125	34.0	29.2	41.2	60.5	24.2	25.5	23.3
gpt-3.5-turbo-1106	33.7	26.8	41.7	51.5	28.6	27.8	26.0
command-r-plus	33.0	20.3	24.6	71.5	23.9	24.9	32.7

Finally, we include the `Plot Unscrambling` task, which takes the plot synopses of recently-released movies from IMDb or Wikipedia. We randomly shuffle the synopses sentences and then ask the LLM to simply reorder the sentences into the original plot. We find that this task is very challenging for LLMs, as it measures their abilities to reason through plausible sequences of events.

3 Experiments

In this section, first we describe our experimental setup and present full results for 43 LLMs on all 18 tasks of `LiveBench`. Next, we give an empirical comparison of `LiveBench` to existing prominent LLM benchmarks, and finally, we present ablation studies.

Experimental setup. Our experiments include 43 LLMs total, with a mix of top proprietary models, large open-source models, and small open-source models. In particular, for proprietary models, we include six GPT models: `gpt-4o-2024-05-13`, `gpt-4-turbo-2024-04-09`, `gpt-4-1106-preview`, `gpt-4-0125-preview`, `gpt-3.5-turbo-1106`, `gpt-3.5-turbo-0125` [6, 41]; three Anthropic models: `claude-3-opus-20240229`, `claude-3-sonnet-20240229`, `claude-3-haiku-20240307` [2]; two Mistral models: `mistral-large-2402`, `mistral-small -2402` [29]; and two Google models: `gemini-1.5-pro-api-0514` and `gemini-1.5-pro-flash-0514` [45].

For large open-source models, we include `command-r`, `command-r-plus` [13], `meta-llama-3-70b-instruct` [38], `mixtral-8x22b-instruct-v0.1`, `mixtral-8x7b -instruct-v0.1` [29], `qwen1.5-110b-chat`, and `qwen1.5-72b-chat` [3].

For small open-source models, we include `llama-2-7b-chat-hf` [56], `llama-3-8b-instruct` [38], `mistral-7b-instruct-v0.2` [29], `phi-3-medium-4k-instruct`, `phi-3-medium-128k-instruct`,

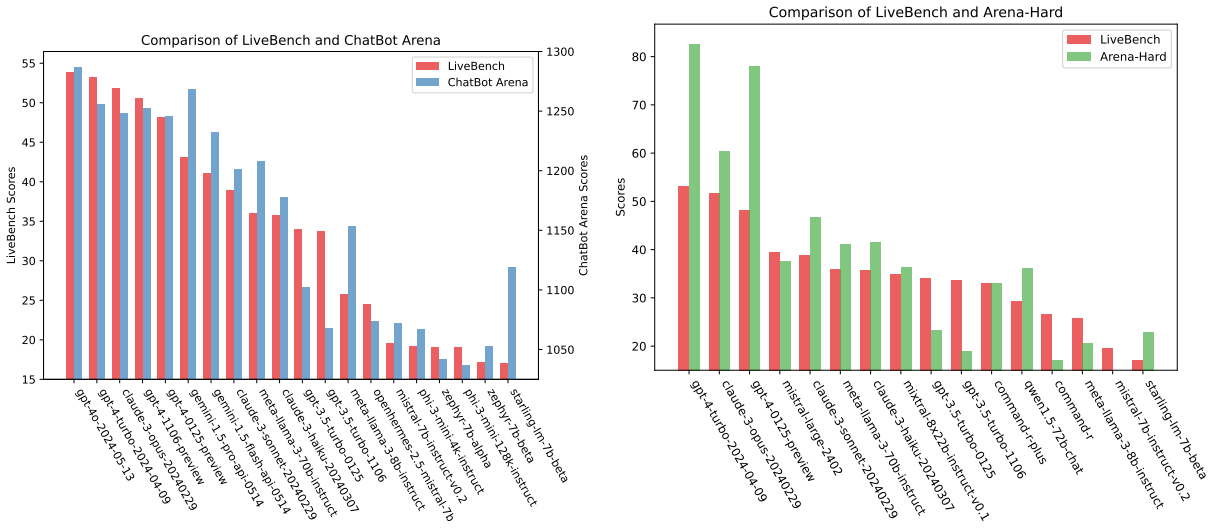

Figure 2: **Comparison of LiveBench to other LLM benchmarks.** We compare `LiveBench` to ChatBot Arena (left) and Arena-Hard (right). We see that while there are generally similar trends, some models are noticeably stronger on one benchmark vs. the other. For example, both GPT-4 models are substantially better on Arena-Hard, likely due to the known bias from using `gpt-4` itself as the LLM judge [34].

`phi-3-small-8k-instruct`, `phi-3-small-128k-instruct`, `phi-3-mini-128k-instruct`, `phi-3-mini-4k-instruct` [1], `qwen1.5-0.5b-chat`, `qwen1.5-1.8b-chat`, `qwen1.5-4b-chat`, `qwen1.5-7b-chat`, `qwen2-0.5b-instruct`, `qwen2-1.5b-instruct`, `qwen2-7b-instruct` [3], `starling-lm-7b-beta` [67], `teknium/openhermes-2.5-mistral-7b` [54], `vicuna-7b-v1.5`, `vicuna-7b-v1.5-16k` [9], `yi-6b-chat` [62], `zephyr-7b-alpha`, and `zephyr-7b-beta` [57].

For all models and tasks, we perform single-turn evaluation with temperature 0. All models run with their respective templates from FastChat [65]. We run all open-source models with `bfloat16`. For each question, a model receives a score from 0 to 1. For each model, we compute the score on each task as the average of all questions, we compute the score on each of the six categories as the average of all their tasks, and we compute the final `LiveBench` score as the average of all six categories.

3.1 Discussion of Results

We compare all 43 models on `LiveBench` according to the experimental setup described above; see Table 1 and Table 3. We find that `gpt-4o-2024-05-13` performs the best overall, with `gpt-4-turbo-2024-04-09` and `claude-3-opus-20240229` not far behind. The best-performing open-source model is `qwen2-72b-instruct`, and `meta-llama-3-8b-instruct` is the best-performing open-source model that is 8B or smaller.

We find that `gpt-4-turbo-2024-04-09` is the highest-performing LLM in the reasoning category, 8% above the next-best model. Furthermore, the GPT-4 models, particularly `gpt-4-turbo-2024-04-09`, are the highest-performing models in coding, which is in line with recent existing results [27, 48]. `gpt-4-turbo-2024-04-09` performs best on average over the 'quantitative reasoning tasks': coding, data analysis, math, and reasoning.

Table 2: **LLM judges cannot accurately evaluate challenging math and reasoning questions.** Error rate of LLM-as-a-judge scoring on challenging math (AMC, AIME, SMC) and reasoning (Zebra puzzles) tasks. On all tasks, the error rate is surprisingly high, showing that LLMs are not reliable judges for these tasks.

Model	Judge	AMC12 2024	AIME 2024	SMC 2023	Zebra Puzzles
GPT-4-Turbo	GPT-4-Turbo	0.380	0.214	0.353	0.420
Claude-3-Opus	GPT-4-Turbo	0.388	0.103	0.294	0.460

3.2 Comparison to Other LLM Benchmarks

Next, we compare `LiveBench` to two prominent benchmarks, ChatBot Arena [10] and Arena-Hard [34]. In Figure 2, we show a bar plot comparison among models that are common to both benchmarks, and in Figure 3, we compare the performance of these models to a best-fit line. We also compute the correlation coefficient of model scores among the benchmarks: `LiveBench` has a 0.92 and 0.90 correlation with ChatBot Arena and Arena-Hard, respectively.

Based on the plots and the correlation coefficients, we see that there are generally similar trends to `LiveBench`, yet some models are noticeably stronger on one benchmark vs. the other. For example, `gpt-4-0125-preview` and `gpt-4-turbo-2024-04-09` perform substantially better on Arena-Hard compared to `LiveBench` – likely due to the known bias from using `gpt-4` itself as the LLM judge [34]. We hypothesize that the strong performance of some models such as `gemini-1.5-pro-latest` and `starling-lm-7b-beta` on ChatBot Arena compared to `LiveBench` may be due to having an output style that is preferred by humans. These observations emphasize the benefit of using ground-truth judging, which is immune to biases based on the style of the output.

3.3 Comparison between Ground-Truth and LLM-Judging

In this section, we run an ablation study to compare the result of ground-truth judging with LLM judging, by taking three math sub-tasks and one reasoning task and scoring them by either matching with the ground-truth answer or by asking an LLM judge to score the answer as either correct or incorrect. We use a judge prompt based on the MT-Bench judge prompt (see Appendix A.1 for details), and we use `gpt-4-turbo-2024-04-09` as the judge. We judge the model outputs of both `gpt-4-turbo-2024-04-09` and `claude-3-opus-20240229` in Table 2. We find that the error rate for all tasks is far above a reasonable value, indicating that LLM judges are not appropriate for challenging math and logic tasks. Interestingly, the lowest error rates are on AIME 2024, which is also the task with the lowest overall success rate according to ground-truth judgment.

4 Related Work

We describe the most prominent LLM benchmarks and the ones that are most related to our work. For a comprehensive survey, see [8]. The Huggingface Open LLM Leaderboard [4, 21] is a widely-used benchmark suite that consists of six static datasets: ARC [11], GSM8K [12], HellaSwag [63], MMLU [25], TruthfulQA [36], and Winogrande [47]. While this has been incredibly useful in tracking the performance of LLMs, its static nature has left it prone to test set contamination by models.

LLMs-as-a-judge. AlpacaEval [17, 18, 35], MT-Bench [10], and Arena-Hard [34] are benchmarks that employ LLM judges on a fixed set of questions. Using an LLM-as-a-judge is fast and relatively cheap. Furthermore, this strategy has the flexibility of being able to evaluate open-ended questions, instruction-following questions, and chatbots. However, LLM judging also has downsides. First, LLMs have biases towards their own answers [34]. In addition to favoring their own answers [34], GPT-4 judges have a noticeable difference in terms of variance and favorability of other models compared to Claude judges. Additionally, LLMs make errors. As one concrete example, question 2 in Arena-Hard asks a model to write a C++ program to compute whether a given string can be converted to 'abc' by swapping two letters. GPT-4 incorrectly judges `gpt-4-0314`'s solution as incorrect [34].

Humans-as-a-judge. ChatBot Arena [10, 65] leverages human prompting and feedback on a large scale. Users ask questions and receive outputs of two randomly selected models and have to pick which output they prefer. This preference feedback is aggregated into Elo scores for the different models. While human evaluation is great for capturing the preferences of a crowd, using a human-as-a-judge has many disadvantages. First, human-judging can be quite labor-intensive, especially for certain tasks included in `LiveBench` such as complex math, coding, or long-context reasoning problems. Whenever humans are involved in annotation (of which judging is a sub-case), design choices or factors can cause high error rates [31], and even in well-designed human-annotation setups, high variability from human to human leads to unpredictable outcomes [43].

Other benchmarks Perhaps the most-related benchmark to ours is LiveCodeBench [27], which also regularly releases new questions and makes use of ground-truth judging. However, it is limited to only coding tasks. Concurrent work, the SEAL Benchmark [48], uses private questions with expert human scorers, however, the benchmark currently only contains the following categories: Math, Coding, Instruction Following, and Spanish. In [49], the authors modify the original MATH dataset [26] by changing numbers in the problem setup. They find drastic declines in model performance for all LLMs including the frontier ones. However, while such work can evaluate LLMs on data that is not in the pretraining set, the data still ends up being highly similar to the kind of data likely seen in the pretraining set. In addition, the hardness of the benchmark remains the same over time.

5 Conclusions, Limitations, and Future Work

In this work, we introduced LiveBench, an LLM benchmark designed to mitigate both test set contamination and the pitfalls of LLM judging and human crowdsourcing. LiveBench is the first benchmark that (1) contains frequently updated questions from new information sources, in which questions become harder over time, (2) scores answers automatically according to objective ground-truth values, without the use of LLM judges, and (3) contains a wide variety of challenging tasks, spanning math, coding, reasoning, language, instruction following, and data analysis. LiveBench contains questions that are based on recently released math competitions, arXiv papers, and datasets, and it contains harder, 'contamination-proof' versions of previously released benchmarks. We released all questions, code, and model answers, and questions will be added and updated on a monthly basis. We welcome community collaboration for expanding the benchmark tasks and models.

Limitations and Future Work. While we attempted to make `LiveBench` as diverse as possible, there are still additions from which it would benefit. For example, we hope to add non-English language tasks in the future. Furthermore, while ground truth scoring is beneficial in many ways, it still cannot be used for certain use cases, such as 'write an email to my boss', or 'write a travel guide to Hawaii' in which it is hard to define a ground truth. Finally, while we attempted to make all tasks and categories fair for all models, there are still biases due to certain LLM families favoring certain prompt types. We plan to update the prompts (at the start and end of each question) in the future, as new prompt strategies are developed. Similarly, we plan to update the `LiveBench` leaderboard as new LLMs are released.

References

[1] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[2] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 2024.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[4] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[5] BIG bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[9] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

[10] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

[11] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Cohere. Command r: Retrieval-augmented generation at production scale. https://txt.cohere.com/command-r, March 2024.

[14] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. *arXiv preprint arXiv:2311.09783*, 2023.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, and Ge Li. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. *arXiv preprint arXiv:2402.15938*, 2024.

[17] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

[18] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

[19] J Douglas Faires and David Wells. *The Contest Problem Book VIII: American Mathematics Competitions (AMC 10) 2000–2007*, volume 19. American Mathematical Society, 2022.

[20] Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models, October 2023. arXiv:2310.18208 [cs].

[21] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021.

[22] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[23] Shahriar Golchin and Mihai Surdeanu. Time travel in llms: Tracing data contamination in large language models. *arXiv preprint arXiv:2308.08493*, 2023.

[24] Guardian Media Group. The guardian. https://www.theguardian.com/, 1821. Accessed: 2024-01-20.

[25] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[26] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[27] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

[28] S Jeremy. Einstein's riddle: Riddles, paradoxes, and conundrums to stretch your mind. *Bloomsbury USA*, pages 10–11, 2009.

[29] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[30] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[31] Matthew Lease. On quality control and machine learning in crowdsourcing. In *Workshops at the twenty-fifth AAAI conference on artificial intelligence*, 2011.

[32] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[33] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

[34] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, April 2024.

[35] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023.

[36] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022.

[37] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[38] Meta. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/, April 2024. Accessed: June 4, 2024.

[39] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[41] OpenAI. Gpt-4 technical report. *Technical Report*, 2023.

[42] quint t. Puzzle generator and puzzle solver. https://github.com/quint-t/Puzzle-Generator-and-Solver, 2023.

[43] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring attribution in natural language generation models. *Computational Linguistics*, 49(4):777–840, 2023.

[44] John W. Ratcliff and David E. Metzener. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, page 46, 1988.

[45] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[46] Manley Roberts, Himanshu Thakur, Christine Herlihy, Colin White, and Samuel Dooley. To the cutoff... and beyond? a longitudinal perspective on llm data contamination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[47] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, Apr. 2020.

[48] Scale AI. Seal leaderboards. https://scale.com/leaderboard, May 2024.

[49] Saurabh Srivastava, Anto PV, Shashank Menon, Ajay Sukumar, Alan Philipose, Stevin Prince, Sooraj Thomas, et al. Functional benchmarks for robust evaluation of reasoning performance, and the reasoning gap. *arXiv preprint arXiv:2402.19450*, 2024.

[50] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.

[51] AtCoder Team. Atcoder. https://atcoder.jp/.

[52] LeetCode Team. Leetcode. https://leetcode.com/.

[53] Python 3 Team. difflib. https://docs.python.org/3/library/difflib.html.

[54] Teknium. teknium/openhermes-2.5-mistral-7b, 2023.

[55] Graham Todd, Tim Merino, Sam Earle, and Julian Togelius. Missed connections: Lateral thinking puzzles for large language models, 2024.

[56] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[57] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. Zephyr: Direct distillation of lm alignment, 2023.

[58] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[59] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[60] Cong Yan and Yeye He. Auto-Suggest: Learning-to-Recommend Data Preparation Steps Using Data Science Notebooks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554, Portland OR USA, June 2020. ACM.

[61] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 36, 2024.

[62] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

[63] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[64] Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, Will Song, Tiffany Zhao, Pranav Raja, Dylan Slack, Qin Lyu, et al. A careful examination of large language model performance on grade school arithmetic. *arXiv preprint arXiv:2405.00332*, 2024.

[65] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[66] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[67] Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaif, November 2023.

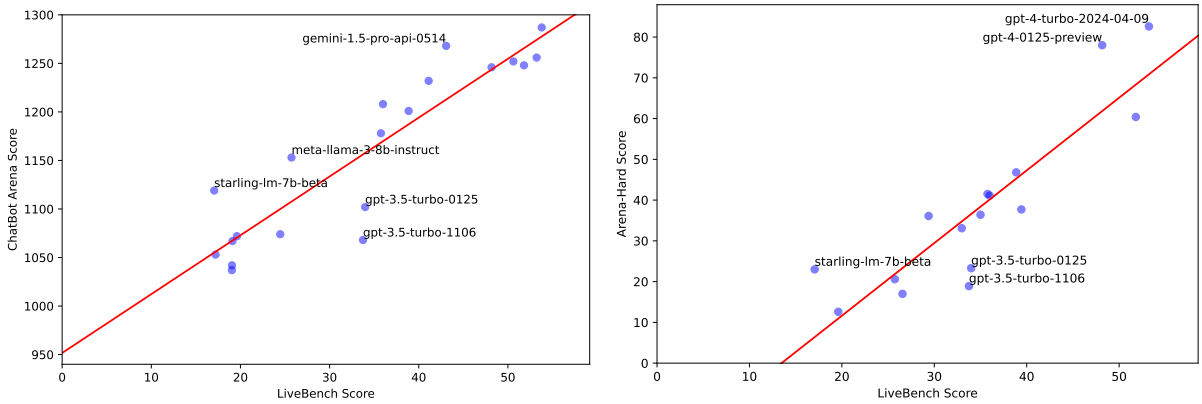

Figure 3: **The performance of models on different benchmarks, compared to a best-fit line.** We compare the different in relative performance of LLMs on `LiveBench` vs. ChatBot Arena, and `LiveBench` vs. Arena-Hard. We see that while many models are near the best-fit lines, a few are notable outliers, providing evidence that their output style may be noticeably better or worse than their ability to answer questions.

A Additional Details about LiveBench Experiments

In this section, we detail further descriptions about the `LiveBench` benchmark itself and our experiments. For example, we include further depictions of the comparisons of `LiveBench` to ChatBot Arena and Arena-Hard in Figure 3. We display the full results table for `LiveBench` in Table 3. We display the list of all verifiable instructions in Table 4. Other details are below.

A.1 Details from Ablation Studies

In this section, we give more details from Section 3.

Recall that in Section 3.3, we ran an ablation study by taking three math sub-tasks and one reasoning task, and scoring them by either matching with the ground truth answer, or by asking an LLM judge to score the answer as either correct or incorrect. We used a judge prompt similar to MT-Bench, which we duplicate below. Furthermore, to complement Table 2, we give the model performance scores for ground-truth and LLM judging for the respective models and tasks, in Table 5.

> [Instruction] Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider correctness alone. Identify and correct any mistakes. Be as objective as possible. After providing your explanation, you must rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]". [Question] question [The Start of Assistant's Answer] answer [The End of Assistant's Answer]

A.2 Detailed Description of LiveBench Categories

In this section, we describe the categories and tasks of `LiveBench` in more detail.

A.2.1 Math Category

Evaluating the mathematical abilities of LLMs has been one of the cornerstones of recent research in LLMs, featuring prominently in many releases and reports [6, 7, 41, 45]. Our benchmark includes math questions of three types: questions from recent high school math competitions, fill-in-the-blank questions from recent proof-based USAMO and IMO problems, and questions from our new, harder version of the AMPS dataset [26].

Math competitions. Our first math category uses expert human-designed math problems that offer a wider variety in terms of problem type and solution technique. We focus on high school math competition questions from English-speaking countries: AMC12, AIME, SMC, and USAMO, and also IMO, the international competition.

First, we include the *American Mathematics Competition 12 (AMC12)*, both AMC12A and AMC12B 2023, released on November 8, 2023 and November 14, 2023, respectively, and the Senior Mathematical Challenge (SMC) 2023, released on October 3, 2023. All three are challenging multiple-choice competitions for high school students in the USA (AMC) and UK (SMC) that build in difficulty, meant as the first step for high school students to qualify for their country's team for the International Mathematical Olympiad (IMO).

The questions test mathematical problem solving with arithmetic, algebra, counting, geometry, number theory, probability, and other secondary school math topics [19]. An example of a problem of this type from the AMC12A 2023 problem set is below:

An example question from the Math Competitions task.
How many complex numbers satisfy the equation $z^5 = \overline{z}$, where $\overline{z}$ is the conjugate of the complex number z? **(A)** 2 **(B)** 3 **(C)** 5 **(D)** 6 **(E)** 7
If you cannot determine the correct multiple-choice answer, take your best guess. Once you have your answer, please duplicate that letter five times in a single string. For example, if the answer is F, then write FFFFF.

Ground Truth: EEEEE

Next, we include the *American Invitational Mathematics Examination (AIME)*, both AIME I and AIME II 2024, released on January 31, 2024 and February 7, 2024, respectively. These are prestigious and challenging tests given to those who rank in the top 5% of the AMC. Each question's answer is an integer from 000 to 999. An example of a problem of this type from the AIME I 2024 problem set is below:

An example question from the Math Competitions task.
Real numbers x and y with $x, y > 1$ satisfy $\log_x(y^x) = \log_y(x^{4y}) = 10$. What is the value of xy? Please think step by step, and then display the answer at the very end of your response. The answer is an integer consisting of exactly 3 digits (including leading zeros), ranging from 000 to 999, inclusive. For example, the answer might be 068 or 972. If you cannot determine the correct answer, take your best guess. Remember to have the three digits as the last part of the response.

Ground Truth: 025

Proof-based questions. Lastly, we consider the *USA Math Olympiad (USAMO)* 2023 and *International Math Olympiad (IMO)* 2023 competitions, released on March 21, 2023 and July 2, 2023,

respectively. These contests are primarily proof-based and non-trivial to evaluate in an automated way. One possibility is to use LLMs to evaluate the correctness of the natural language proof. However, we then have *no* formal guarantees on the correctness of the evaluation. Another possibility is to *auto-formalize* the proofs into a formal language such as Lean and then run a proof checker. However, while there have been notable recent improvements in auto-formalization, such a process still does not have formal guarantees on the correctness of the auto-formalization – and thus that of the evaluation. To tackle this, we formulate a novel task which can test the ability of an LLM in the context of proofs. Specifically, for a proof, we *mask* out a subset of the formulae in the proof. We then present the masked out formulae in a *scrambled* order to the LLM and ask it to *reinsert* the formulae in the correct positions. Such a task tests the mathematical, deductive, and instruction following abilities of the LLM. In particular, if the LLM is strong enough to generate the correct proof for a question, then one would expect it to also solve the far easier task of completing a proof which has some missing formulae – especially if the formulae are already given to it in a scrambled order. Note that this also allows us to easily control the level of difficulty of the question by changing the number of formulae that we mask.

We generate 3 hardness variants for each problem, masking out 10%, 50% and 80% of the equations in the proof. We evaluate by computing the edit distance between the ground truth ranking order and the model predicted ranking order. [NB : in preliminary testing we also evaluated using the accuracy metric and the model rankings remained nearly the same]. Models perform worse on IMO compared to USAMO, in line with expectations. We also looked at the performance as separated by question hardness. The scores are greatly affected by question hardness going from as high as 96.8 for the easiest questions (10% masked out, GPT-4o) to as low as 36 for the hardest (80% masked out). The full results are in Table 6 and Table 7.

An example of a problem of this type from the IMO problem set can be found here.

Synthetically generated math questions. Finally, we release synthetic generated math questions. This technique is inspired from math question generation used to create the MATH and AMPS datasets [26]. In particular, we randomly generate a math problem of one of several types, such as taking the derivative or integral of a function, completing the square, or factoring a polynomial. We generate questions by drawing random primitives, using a larger (and therefore more challenging) distribution than AMPS. Note that, for problem types such as integration, this simple technique of drawing a random function and taking its derivative results in a wide variety of integration problems of varying difficulty. For example, problem solutions may involve applying the chain rule, the product/quotient rule, trigonometric identities, or use a change of variables. In order to extract the answer, we ask the model to use the same 'latex boxed answer' technique as in the MATH dataset [26]. We judge the correctness of answers as in the EleutherAI Eval Harness [21] using Sympy [39] where we check for semantic as well as numerical equivalence of mathematical expressions. An example of a integral problem is as follows:

An example question from the AMPS Hard task.
Find an indefinite integral (which can vary by a constant) of the following function: $5\sec^2(5x+1) - 8\sin(7-8x)$. Please put your final answer in a *boxed*{}.

Ground Truth: $-\sin(7)\sin(8x) - \cos(7)\cos(8x) + \tan(5x+1)$

A.2.2 Coding Category

The coding ability of LLMs is one of the most widely studied and sought-after skills for LLMs [27, 33, 40]. We include two coding tasks in `LiveBench`: a modified version of the code generation task from LiveCodeBench [27], and a novel code completion task combining LiveCodeBench problems with partial solutions collected from GitHub sources. Examples of questions from the Coding tasks can be found here.

Code generation. In the `LCB Generation` task, we assess a model's ability to parse a competition coding question statement and write a correct answer. LiveCodeBench [27] included several tasks to assess the coding capabilities of large language models. We have taken 50 randomly selected problems from the April 2024 release of LiveCodeBench, selecting only problems released in or after November 2023. The problems are competition programming problems from LeetCode [52] and AtCoder[51], defined with a textual description and solved by writing full programs in Python 3 code.

These problems are presented as in LiveCodeBench's Code Generation task, with minor prompting differences and with only one chance at generating a correct solution per question, per model. We report pass@1, a metric which describes the proportion of questions that a given model solved completely (a solution is considered correct if and only if it passes all public and private test cases).

Code completion. In this task, we assess the ability of the model to successfully complete a partially provided solution to a competition coding question statement. The setup is similar to the Code Generation task above, but a partial (correct) solution is provided in the prompt and the model is instructed to complete it to solve the question. We use LeetCode medium and hard problems from LiveCodeBench's [27] April 2024 release, combined with matching solutions from `https://github.com/kamyu104/LeetCode-Solutions`, omitting the last 15% of each solution and asking the LLM to complete the solution. As with Code Generation, we report pass@1.

A.2.3 Reasoning Category

The reasoning abilities of large language models is another highly-benchmarked and analyzed skill of LLMs [50, 58, 61]. In `LiveBench`, we include three reasoning tasks: harder versions of tasks from Big-Bench Hard [50] and bAbI [59], and Zebra puzzles.

Web of lies v2. Web of Lies is a task included in Big-Bench [5] and Big-Bench Hard [50]. The task is to evaluate the truth value of a random Boolean function expressed as a natural-language word problem. In particular, the LLM must evaluate $f_n(f_{n-1}(...f_1(x)...))$, where each f_i is either negation or identity, and x is True or False. We represent x by the sentence: X_0 {tells the truth, lies}, and we represent f_i by a sentence: X_i says X_{i-1} {tells the truth, lies}. The sentences can be presented in a random order for increased difficulty. For example, a simple $n = 2$ version is as follows: 'Ka says Yoland tells the truth. Yoland lies. Does Ka tell the truth?' Already by October 2022, LLMs achieved near 100% on this task, and furthermore, there are concerns that Big-Bench tasks leaked into the training data of GPT-4, despite using canary strings [41].

For `LiveBench`, we create a new, significantly harder version of Web of Lies. We make the task harder with a few additions: *(1)* adding different types of red herrings, *(2)* asking for the truth values of three people, instead of just one person, and *(3)* adding a simple additional deductive

component. For *(1)*, we maintain a list of red herring names, so that the red herrings do not affect the logic of the answer while still potentially leading LLMs astray. For example, 'Fred says Kayla lies,' where Fred is in the true 'web of lies', while Kayla may lead to a series of steps ending in a dead end. Overall, the number of total red herring sentences is drawn from a uniform distribution ranging from 0 to 19. For *(3)*, we simply assign each name to a location and give sentences of the form 'Devika is at the museum. The person at the museum says the person at the ice skating rink lies.' We find that this makes the task significantly harder for leading LLMs, even without shuffling the sentences into a random order.

An example question from the Web of Lies v2 task.
In this question, assume each person either always tells the truth or always lies. Tala is at the movie theater. The person at the restaurant says the person at the aquarium lies. Ayaan is at the aquarium. Ryan is at the botanical garden. The person at the park says the person at the art gallery lies. The person at the museum tells the truth. Zara is at the museum. Jake is at the art gallery. The person at the art gallery says the person at the theater lies. Beatriz is at the park. The person at the movie theater says the person at the train station lies. Nadia is at the campground. The person at the campground says the person at the art gallery tells the truth. The person at the theater lies. The person at the amusement park says the person at the aquarium tells the truth. Grace is at the restaurant. The person at the aquarium thinks their friend is lying. Nia is at the theater. Kehinde is at the train station. The person at the theater thinks their friend is lying. The person at the botanical garden says the person at the train station tells the truth. The person at the aquarium says the person at the campground tells the truth. The person at the aquarium saw a firetruck. The person at the train station says the person at the amusement park lies. Mateo is at the amusement park. Does the person at the train station tell the truth? Does the person at the amusement park tell the truth? Does the person at the aquarium tell the truth? Think step by step, and then put your answer in **bold** as a list of three words, yes or no (for example, **yes, no, yes**). If you don't know, guess.

Ground Truth: no, yes, yes

House Traversal. Next, we include a harder version of a task from bAbI [59]. bAbI is a dataset consisting of reasoning tasks in natural language. One of the tasks is a spatial reasoning task, consisting of sentences of the form, 'The bedroom is West of the kitchen. The kitchen is South of the garden' and asking the model where a room is in relation to another room. We make this task harder by adding a distinct person in each room, and asking the LLM who a person would see if they traverse in a few directions. We find that this task is very challenging to LLMs *even with just a 2x2 layout of rooms.*

An example question from the House Traversal task.
The office is directly West of the living room. The library is directly West of the laundry room. The office is directly North of the library. Layla is in the office. Chloe is in the living room. Diego is in the library. Meera is in the laundry room. Chloe went one room West. Who did Chloe see, and in what order? Think step by step, and then give your answer as a list of names in **bold**. If you don't know, guess.

Ground Truth: Layla

Zebra puzzles. The final reasoning task we include is Zebra puzzles. Zebra puzzles, also called Einstein's riddles or Einstein's puzzles, are a well-known [28] reasoning task that tests the ability of the model to follow a set of statements that set up constraints, and then logically deduce the requested information. The following is an example with three people and three attributes:

An example question from the Zebra Puzzle task.
There are 3 people standing in a line numbered 1 through 3 in a left to right order.
Each person has a set of attributes: Food, Nationality, Hobby.
The attributes have the following possible values:

- Food: nectarine, garlic, cucumber
- Nationality: chinese, japanese, thai
- Hobby: magic-tricks, filmmaking, puzzles

and exactly one person in the line has a given value for an attribute.
Given the following premises about the line of people:

- the person that likes garlic is on the far left
- the person who is thai is somewhere to the right of the person who likes magic-tricks
- the person who is chinese is somewhere between the person that likes cucumber and the person who likes puzzles

Answer the following question:
What is the hobby of the person who is thai? Return your answer as a single word, in the following format: ***X***, where X is the answer.

Ground Truth: filmmaking

We build on an existing repository for procedural generation of Zebra puzzles [42]; the repository allows for randomizing the number of people, the number of attributes, and the set of constraint statements provided. For the attribute randomization, they are drawn from a set of 10 possible categories (such as Nationality, Food, Transport, Sport) and for each of these categories there are between 15 and 40 possible values to be taken. For the constraint statements, the implementation allows for up to 20 'levels' of constraint in ascending order of intended difficulty. For example, level 1 could include a statement such as 'The person who likes garlic is on the left of the person who plays badminton' and a level 10 statement could be 'The person that watches zombie movies likes apples or the person that watches zombie movies likes drawing, but not both'. Higher levels also include lower level statements in their possible set of statements to draw from, but this set narrows progressively as the level increases from 12 to 20 by removing the possibility of having lower-level statements (starting with removing level 1, then removing level 2, etc).

The repository also includes a solver for the puzzles, which we use to ensure there is a (unique) solution to all of our generated puzzles.

Our modifications to the original repository primarily target the reduction of ambiguity in the statements (e.g. changing 'X is to the left of Y' to 'X is to the *immediate* left of Y'). For generation, we pick either 3 or 4 people with 50% probability, either 3 or 4 attributes with 50% probability, and we draw the levels from the integer interval [10, 20] with uniform probability. In preliminary testing, we found that larger puzzles proved exceedingly difficult for even the top performing LLMs.

A.2.4 Data Analysis

While LLMs abilities in the previous three categories have been widely studied, we also include a category that is significantly less-studied but is a practical application of LLMs which requires more attention: data analysis tasks. We include three tasks in which the LLM assists in data analysis or data science: column type annotation, table join prediction, and table reformatting.

All problems are based on recent datasets from Kaggle and Socrata. Owing to the limited output context lengths of the current generation of LLMs and the comparatively high per-token costs of generating responses, we upper bound the size of our tables with respect to cell length, column count and row count. We note that this is a limitation of all current-generation LLMs up to and including state-of-the-art models.

Example questions from the Data Analysis category can be lengthy, so examples can be viewed here.

Column type annotation. Consider a table A with t columns and r rows. We denote each column $C \in A$ as a function which maps row indices to strings; i.e., for $0 \leq i < t$, we have $C_i : \mathbb{N} \to \Sigma_*$, where i is the column index. Let $L \subseteq \Sigma_*$ denote a label set; these are our column types to be annotated. Standard CTA assumes a fixed cardinality for this label set, indexed by a variable we call j. Given the above definitions, we define single-label $CTA \subset A \times L$ as a relation between tables and labels:

$$\forall C, \exists l_j \mid (C_i, l_j) \in CTA \tag{1}$$

We seek a generative method $M : \Sigma_* \to \Sigma_*$ that comes closest to satisfying the following properties:

$$M(\sigma, L) \in L, \forall C \in A, M(\sigma, L) \in CTA \tag{2}$$

For further details on the task, please refer to [20]. **Implementation details.** For each benchmark instance, we retrieve a random A from our available pool of recent tables. We randomly and uniformly sample C from A, use the actual column name of A as our CTA ground-truth L, and retrieve $\sigma_1 \cdots \sigma_5$ column samples from C, with replacement, providing them as context for the LLM. **Metrics.** We report Accuracy @ 1 over all instances, accepting only case-insensitive exact string matches as correct answers.

Table reformatting. Given a table A rendered according to a plaintext-readable and valid schema for storing tabular information a_s, we instruct the LLM to output the same table with the contents unchanged but the schema modified to a distinct plaintext-readable valid schema b_s. **Implementation details.** We use the popular library Pandas to perform all of our conversions to and from text strings. We allow the following formats for both input and output: "JSON", "JSONL", "Markdown", "CSV", "TSV", "HTML". As tabular conversion from JSON to Pandas is not standardized, we accept several variations. At inference time, we ingest the LLM response table directly into Pandas. **Metrics.** We report Accuracy @ 1 over all instances. An instance is accepted only if it passes all tests (we compare column count, row count, and exact match on row contents for each instance).

Join-column prediction. Given two tables A and B, with columns $a_1, \ldots$ and $b_1, \ldots$ respectively, the *join-column prediction* task is to suggest a pair (a_k, b_l) of columns such that the equality condition $a_k = b_l$ can be used to join the the tables in a way that matches with the provided

ground-truth mapping $M : A \to B$. The mapping is usually partial injective: not every column in B is mapped from A, not every column in A is mapped to B. For further details, please refer to [60]. **Implementation details.** We randomly sample columns with replacement from our entire collection of tables, generating a fixed column pool C. We retain half the rows of A to provide as context to the LLM. The remaining rows are used to generate a new table B. For each instance, we randomly sample columns from both the target table and the column pool and join them to B. We anonymize the column names in B, then pass both A and B to the LLM and ask it to return a valid join mapping M. **Metrics.** We report the F1 score over columns, with TPs scored as exact matches between ground truth and the LLM output, FPs scored as extraneous mappings, FNs scored as missing mappings, and incorrect mappings counting as FP + FN.

A.2.5 Instruction Following

An important ability of an LLM is its capability to follow instructions. To this end, we include instruction following questions in our benchmark, inspired by IFEval [66].

Generating live prompts and instruction. IFEval, or instruction-following evaluation for LLMs, contains verifiable instructions such as "write more than 300 words" or "Finish your response with this exact phrase: {end_phrase}." These instructions are then appended to prompts like "write a short blog about the a visit in Japan". We use this modular nature between the prompt and instruction to construct live prompts.

For our live source, we considered news articles from The Guardian; we are able to obtain 200 articles using their API[1]. Using the first n sentences article text as the source text, we consider four different tasks using the text: paraphrase, summarize, simplify, and story generation. The exact prompts can be seen in Table 8. For the instructions, we use the code provided by [66], making a few modifications such as increasing the max number of keywords from two to five. Additionally, we compose different instructions together by sampling from a uniform distribution from 2 to 5. However, since the instructions can be conflicting, we deconflict the instructions. This results in approximate normal distribution of the number of instructions per example with the majority of the containing two or three instructions. A full list of the instructions can be found in Appendix Table 4. To construct, the full prompt, containing the news article sentences, the prompt, and the instructions, we use the following meta prompt: "The following are the beginning sentences of a news article from the Guardian.\n———-\n{guardian article}\n———-\n{subtask prompt} {instructions}".

Scoring. To evaluate the model's performance on instruction following, we use a scoring method that considers two key factors: whether all instructions were correctly followed for a given prompt, i.e. Prompt-level accuracy, and what fraction of the individual instructions were properly handled, i.e. Instruction-level accuracy. The first component of the score checks if the model successfully followed every instruction in the prompt and assigns 1 or 0 if it missed any of the instructions. The second component looks at each individual instruction and checks whether it was properly followed or not. The final score is the average of these two components, scaled to lie between 0 and 1. A score of 1 represents perfect adherence to all instructions, while lower scores indicate varying degrees of failure in following the given instructions accurately.

[1] https://open-platform.theguardian.com/

Example questions from the Instruction Following category can be lengthy, so examples can be viewed here.

A.2.6 Language Comprehension

Finally, we include multiple language comprehension tasks. These tasks assess the language model's ability to reason about language itself by, (1) completing word puzzles, (2) fixing misspellings while leaving other stylistic changes in place, and (3) reordering scrambled plots of unknown movies.

Connections. First we include the 'Connections' category[2]. Connections is a word puzzle category introduced by the New York Times (although similar ideas have existed previously). Sixteen words are provided in a random order; the objective of the game is to sort these into four sets of four words, such that each set has a 'connection' between them. Such connections could include the words belonging to a related category, e.g., 'kiwi, grape, pear, peach' (types of fruits); the words being anagrams, the words being homophones, or being words that finish a certain context, e.g., 'ant, drill, island, opal' being words that come after the word 'fire' to make a phrase. Due to the variety of possible connection types that can exist, the wider knowledge required to understand some connections, as well as some words potentially being 'red herrings' for connections, this task is challenging for LLMs – prior work [55] has comprehensively tested the task on the GPT family of models, as well as on sentence embedding models derived from, e.g., BERT [15] and RoBERTa [37]. The authors found that GPT-4 has an overall completion rate below 40% on the puzzles (when allowed multiple tries to get it correct), concluding that 'large language models in the GPT family are able to solve these puzzles with moderate reliability, indicating that the task is possible but remains a formidable challenge.' In our work, we assess the single-turn performance and test performance on a much larger set of models.

The original task provided for a number of 'retry' attempts in the event of an incorrect submission for a category. To fit into the framework of our benchmark we take the model's answer from a single turn; to ameliorate the increased difficulty of this setting, we use fewer words/groups for some questions. The split we use is 15 questions of eight words, 15 questions of twelve words and 20 questions of sixteen words. An example prompt is as follows:

An example question from the `Connections` task.
You are given 8 words/phrases below. Find two groups of four items that share something in common. Here are a few examples of groups: bass, flounder, salmon, trout (all four are fish); ant, drill, island, opal (all four are two-word phrases that start with 'fire'); are, why, bee, queue (all four are homophones of letters); sea, sister, sin, wonder (all four are members of a septet). Categories will be more specific than e.g., '5-letter-words', 'names', or 'verbs'. There is exactly one solution. Think step-by-step, and then give your answer in **bold** as a list of the 8 items separated by commas, ordered by group (for example, **bass, founder, salmon, trout, ant, drill, island, opal**). If you don't know the answer, make your best guess. The items are: row, drift, curl, tide, current, press, fly, wave.

Ground Truth: current, drift, tide, wave, curl, fly, press, row

The score for this task is the fraction of groups that the model outputs correctly.

[2] See https://www.nytimes.com/games/connections.

Typo Corrections Next, we include details about the `Typos` task. The idea behind this task is inspired by the common use-case for LLMs where a user will ask the system to identify typos and misspellings in some written text. The challenge for the systems is to fix just the typos or misspellings, but to leave other aspects of the text unchanged. It is common for the LLM to impose its own writing style onto that of the input text, such as switching from British to US spellings or adding the serial comma, which may not be desirable.

To create the questions for this task, we take text from recent ArXiv abstracts. These abstracts may themselves start with misspellings and grammatical errors. Therefore, our first step is to manually pass over the abstracts and fix typos and grammar issues. Next, we assemble a list of common misspellings as found online. This is done so as to replicate common misspellings performed by humans, even though we synthetically generate the questions. Finally, for each question, we sample a probability $p \sim U(0.5, 0.7)$ of flipping correctly spelled words to misspelled words. We then use that probability to replace every correctly spelled word with a common misspelling with that probability p. This allows there to be variability in the difficulty of the problem included in the benchmark. In our first release, we include 50 questions. Finally, to score this problem, we merely ask whether the ground truth abstract is contained in the output provided by the LLM.

An example question from the `Typos` task.
Please output this exact text, with no changes at all except for fixing the misspellings. Please leave all other stylistic decisions like commas and US vs British spellings as in the original text.
We introducehten consept of a k-token signed graph adn studdy some of its combinatorial and algebraical properties. We prove that twpo switching isomorphic signed graphs ahve switching isomorphic token graphs. Moreover, we sohw tyhat the Laplacian spectum of a balanced signed graph is contained in the Laplacian spectra of its k-token signed graph. Besides, we introduce and studdyther unbalance levle of a signed graph, which is a new parameter tyhat measures how far a signed graph is frome beng balanced. Moreover, we study the relation bewteen the frustration index anbdther unballance level of signed graphs adn their token signed graphs.

Ground Truth: We introduce the concept of a k-token signed graph and study some of its combinatorial and algebraic properties. We prove that two switching isomorphic signed graphs have switching isomorphic token graphs. Moreover, we show that the Laplacian spectrum of a balanced signed graph is contained in the Laplacian spectra of its k-token signed graph. Besides, we introduce and study the unbalance level of a signed graph, which is a new parameter that measures how far a signed graph is from being balanced. Moreover, we study the relation between the frustration index and the unbalance level of signed graphs and their token signed graphs.

Plot unscrambling. Finally, we include a movie synopsis unscrambling task. We obtain movie plot synopses from IMDb or Wikipedia for feature-length films released after January 1st 2024. These synopses are then split into their constituent sentences and are randomly shuffled. The lengths of the synopses vary from as few as 7 sentences to as many as 66 sentences; at the upper end, this is a very challenging task. The LLM is provided the shuffled sentences with the prompt: 'The following plot summary of a movie has had the sentences randomly reordered. Rewrite the plot summary with the sentences correctly ordered. Begin the plot summary with <PLOT_SUMMARY>.'.

Scoring the task involves two decision points: 1) how to deal with transcription errors - those in which the model modifies the lines when producing its output 2) given the ground truth ordering of

sentences and the LLM's ordering, choosing an appropriate scoring metric. For 1), one option is to permit only strict matching – that is, the LLM must transcribe perfectly. However, although the strongest models do perform well on this (we find they achieve over 95% transcription accuracy), we find that LLMs often correct grammatical errors or spelling mistakes in the source data when transcribing. As we are primarily interested in testing the models' capabilities for *causal language reasoning* in this task, rather than precise transcription accuracy, we instead apply a fuzzy-match using difflib [53] to determine the closest match using a version of the Ratcliff/Obershelp algorithm [44]. For 2), we calculate the score as $1 - \frac{d}{\text{n\_sentences\_gt}}$, where n_sentences_gt is the number of sentences in the ground truth synopsis, and d is the Levenshtein distance [32] of the model's sentence ordering to the ground truth synopsis ordering. Thus if the model's sentence ordering perfectly matches the ground truth, the distance d would be 0, and the score would be 1 for that sample.

One might think that it is plausible that synopsis unscrambling cannot always be solved with the information provided. However, note that even if the set of sentences do not create a distinct causal ordering, the task is essentially asking the LLM to maximize the probability that a given arrangement of sentences is a real movie. In addition to causal reasoning, the LLM can use subtle cues to reason about what ordering creates the most compelling plot. Furthermore, even if there does exist an upper bound on the score that can be achieved that is strictly below 100%, it can still be a useful metric for distinguishing models' relative strengths. An analogous metric is that of next-token perplexity in language modelling; although it is likely that a perfect prediction of the next token is impossible to achieve, and we do not even know what the obtainable lower bound on perplexity is, it is still a powerful metric for determining language-modelling performance.

Example questions from the plot unscrambling task can be lengthy, so examples can be viewed here.

B Additional Documentation

In this section, we give additional documentation for our benchmark. For the full details, see https://github.com/livebench/livebench.

B.1 Author responsibility and license

We, the authors, bear all responsibility in case of violation of rights. The license of our repository is the **Apache License 2.0**.

B.2 Maintenance plan

The benchmark is available on HuggingFace at https://huggingface.co/livebench.

We plan to actively maintain and update the benchmark, and we welcome contributions from the community.

B.3 Code of conduct

Our Code of Conduct is from the Contributor Covenant, version 2.0. See
https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.
The policy is copied below.

> "We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation."

B.4 Datasheet

We include a datasheet [22] for LiveBench at https://github.com/livebench/livebench/blob/main/docs/DATASHEET.md.

B.5 Benchmark statistics

Here, we give statistics on the number of questions and average number of output tokens per task, and the total cost of running `LiveBench` with common API models. For the number of questions for each task, as well as the mean and std. dev number of output tokens per question for `gpt-4-turbo-2024-04-09`, see Table 9. Across 960 questions, with a mean number 412.23 tokens per question, the total cost for running `gpt-4o`, `gpt-4-turbo`, and `gpt-4` is $5.93, $11.87, and $23.74, respectively. Note this only considers the cost of the output tokens, not the input tokens (which are a fraction of the cost of the output). The cost for running `claude-3-opus`, `claude-3-sonnet`, and `claude-3-haiku` is $29.68, $5.93, and $0.49, respectively.

Table 3: **LiveBench Results across 43 models.** We run 14 proprietary and 20 open-source models on `LiveBench`, outputting the results on each main category, as well as each model's overall performance.

Model	LiveBench Score	Coding	Data Analysis	Instruction Following	Language	Math	Reasoning
gpt-4o-2024-05-13	53.8	46.4	52.4	**72.2**	**53.9**	**49.9**	48.0
gpt-4-turbo-2024-04-09	53.2	**47.1**	51.3	71.4	45.3	49.0	**55.3**
claude-3-opus-20240229	51.8	40.1	**54.3**	70.9	51.7	46.5	47.3
gpt-4-1106-preview	50.6	44.4	51.3	69.4	48.4	47.6	42.7
gpt-4-0125-preview	48.2	44.1	54.1	63.9	43.6	42.7	40.7
gemini-1.5-pro-api-0514	43.1	32.8	52.8	67.2	38.3	42.1	25.3
gemini-1.5-flash-api-0514	41.1	39.1	44.0	63.0	30.7	38.5	31.3
mistral-large-2402	39.4	26.8	42.6	68.2	28.7	32.2	38.0
claude-3-sonnet-20240229	38.9	25.2	44.6	65.0	38.1	29.6	30.7
qwen2-72b-instruct	38.5	31.8	26.2	68.3	29.2	43.4	32.0
meta-llama-3-70b-instruct	36.0	20.9	42.4	63.5	34.1	32.3	22.7
claude-3-haiku-20240307	35.8	24.5	41.5	64.0	30.1	25.7	28.7
mixtral-8x22b-instruct-v0.1	35.0	33.1	30.3	63.2	26.5	26.9	30.0
gpt-3.5-turbo-0125	34.0	29.2	41.2	60.5	24.2	25.5	23.3
gpt-3.5-turbo-1106	33.7	26.8	41.7	51.5	28.6	27.8	26.0
command-r-plus	33.0	20.3	24.6	71.5	23.9	24.9	32.7
mistral-small-2402	32.7	24.2	31.9	63.9	22.1	26.8	27.3
phi-3-medium-4k-instruct	29.5	20.6	31.6	53.3	13.9	27.5	30.0
qwen1.5-72b-chat	29.4	22.9	33.0	58.2	11.4	26.8	24.0
phi-3-medium-128k-instruct	29.1	21.6	32.1	56.2	12.8	24.3	28.0
qwen1.5-110b-chat	28.7	22.2	31.5	55.3	13.2	25.6	24.7
phi-3-small-128k-instruct	26.9	25.8	27.3	36.9	12.3	24.8	34.0
command-r	26.6	14.9	31.7	57.2	14.6	16.9	24.0
phi-3-small-8k-instruct	26.5	19.6	27.5	48.2	15.0	24.1	24.7
qwen2-7b-instruct	26.2	29.2	28.7	44.7	10.2	25.8	18.7
meta-llama-3-8b-instruct	25.7	18.3	23.3	57.1	18.7	17.6	19.3
openhermes-2.5-mistral-7b	24.5	11.6	26.9	52.8	11.4	20.1	24.0
mixtral-8x7b-instruct-v0.1	23.1	11.3	28.1	44.8	13.8	19.0	21.3
mistral-7b-instruct-v0.2	19.6	11.6	14.6	51.6	9.1	16.0	14.7
phi-3-mini-4k-instruct	19.1	14.9	14.7	40.1	7.1	19.9	18.0
zephyr-7b-alpha	19.1	11.3	17.4	52.8	7.2	9.6	16.0
phi-3-mini-128k-instruct	19.0	11.6	8.7	49.6	6.8	21.5	16.0
zephyr-7b-beta	17.2	8.3	15.7	48.3	4.3	11.2	15.3
starling-lm-7b-beta	17.1	18.3	2.0	38.3	7.3	13.8	22.7
qwen1.5-7b-chat	16.8	6.6	16.2	44.1	6.2	12.9	14.7
vicuna-7b-v1.5-16k	13.0	1.3	9.3	42.1	7.9	6.6	10.7
vicuna-7b-v1.5	11.2	1.0	2.7	41.8	8.7	4.3	8.7
qwen1.5-4b-chat	10.5	4.0	9.1	27.7	5.8	6.7	9.3
llama-2-7b-chat-hf	10.0	0.0	0.0	44.9	6.9	4.8	3.3
qwen2-1.5b-instruct	9.5	5.6	10.0	25.9	3.0	7.2	5.3
yi-6b-chat	8.3	1.3	4.4	27.2	4.7	7.1	5.3
qwen2-0.5b-instruct	6.6	2.0	2.0	26.6	2.8	4.2	2.0
qwen1.5-1.8b-chat	5.8	0.0	3.3	22.9	3.2	2.1	3.3
qwen1.5-0.5b-chat	5.0	0.0	0.0	21.3	2.9	3.4	2.7

Table 4: The list of 25 instructions used in [66], and the 16 that are both 'real-world' and automatically verifiable, which we used in `LiveBench`. Descriptions are from [66].

Instruction Group	Instruction	Description	In IFEval	In LiveBench
Keywords	Include Keywords	Include keywords {keyword1}, {keyword2} in your response	✓	✓
Keywords	Keyword Frequency	In your response, the word word should appear {N} times.	✓	
Keywords	Forbidden Words	Do not include keywords {forbidden words} in the response.	✓	✓
Keywords	Letter Frequency	In your response, the letter {letter} should appear {N} times.	✓	
Language	Response Language	Your ENTIRE response should be in {language}, no other language is allowed.	✓	
Length Constraints	Number Paragraphs	Your response should contain {N} paragraphs. You separate paragraphs using the markdown divider: * * *	✓	✓
Length Constraints	Number Words	Answer with at least / around / at most {N} words.	✓	✓
Length Constraints	Number Sentences	Answer with at least / around / at most {N} sentences.	✓	✓
Length Constraints	Number Paragraphs + First Word in i-th Paragraph	There should be {N} paragraphs. Paragraphs and only paragraphs are separated with each other by two line breaks. The {i}-th paragraph must start with word {first_word}.	✓	✓
Detectable Content	Postscript	At the end of your response, please explicitly add a postscript starting with {postscript marker}	✓	✓
Detectable Content	Number Placeholder	The response must contain at least {N} placeholders represented by square brackets, such as [address].	✓	
Detectable Format	Number Bullets	Your answer must contain exactly {N} bullet points. Use the markdown bullet points such as: * This is a point.	✓	✓
Detectable Format	Title	Your answer must contain a title, wrapped in double angular brackets, such as <<poem of joy>>.	✓	✓
Detectable Format	Choose From	Answer with one of the following options: {options}	✓	
Detectable Format	Minimum Number Highlighted Section	Highlight at least {N} sections in your answer with markdown, i.e. *highlighted section*	✓	
Detectable Format	Multiple Sections	Your response must have {N} sections. Mark the beginning of each section with {section_splitter} X.	✓	✓
Detectable Format	JSON Format	Entire output should be wrapped in JSON format.	✓	✓
Combination	Repeat Prompt	First, repeat the request without change, then give your answer (do not say anything before repeating the request; the request you need to repeat does not include this sentence)	✓	✓
Combination	Two Responses	Give two different responses. Responses and only responses should be separated by 6 asterisk symbols: ******.	✓	✓
Change Cases	All Uppercase	Your entire response should be in English, capital letters only.	✓	
Change Cases	All Lowercase	Your entire response should be in English, and in all lowercase letters. No capital letters are allowed.	✓	
Change Cases	Frequency of All-capital Words	In your response, words with all capital letters should appear at least / around / at most {N} times.	✓	
Start with / End with	End Checker	Finish your response with this exact phrase {end_phrase}. No other words should follow this phrase.	✓	✓
Start with / End with	Quotation	Wrap your entire response with double quotation marks.	✓	✓
Punctuation	No Commas	In your entire response, refrain from the use of any commas.	✓	

Table 5: Model Performance on math and reasoning tasks with both ground-truth (GT) or LLM judging (LLM-Jdg.)

	AMC12 2024		AIME 2024		SMC 2023		Zebra Puzzles	
	GT	LLM-Jdg.	GT	LLM-Jdg.	GT	LLM-Jdg.	GT	LLM-Jdg.
GPT-4-Turbo	54	64.000	13.793	35.714	70.588	58.824	38	68
Claude-3-Opus	56	42.857	6.897	17.241	58.824	52.941	34	52

Table 6: **IMO/USAMO results for each of 34 models across all hardness levels.**

Model	IMO	USAMO	Avg.
gpt-4o-2024-05-13	60.24	67.47	63.85
gpt-4-1106-preview	58.16	67.17	62.66
claude-3-opus-20240229	52.56	63.66	58.11
gpt-4-turbo-2024-04-09	50.96	64.80	57.88
gemini-1.5-pro-latest	52.11	59.15	55.63
gpt-4-0125-preview	43.04	60.66	51.85
Meta-Llama-3-70B-Instruct	43.24	59.55	51.40
claude-3-sonnet-20240229	44.78	52.97	48.87
command-r-plus	48.33	44.55	46.44
gpt-3.5-turbo-1106	40.37	49.65	45.01
mistral-large-2402	38.65	50.41	44.53
claude-3-haiku-20240307	41.51	47.31	44.41
gpt-3.5-turbo-0125	38.44	47.17	42.80
Qwen1.5-72B-Chat	34.35	48.47	41.41
Mixtral-8x22B-Instruct-v0.1	33.00	48.62	40.81
mistral-small-2402	34.51	44.78	39.64
Meta-Llama-3-8B-Instruct	36.05	36.59	36.32
Qwen1.5-110B-Chat	23.93	46.78	35.35
Mistral-7B-Instruct-v0.2	36.00	34.31	35.15
command-r	31.36	29.38	30.37
Phi-3-mini-128k-instruct	25.84	33.54	29.69
Mixtral-8x7B-Instruct-v0.1	26.52	32.50	29.51
Phi-3-mini-4k-instruct	26.60	30.33	28.46
Qwen1.5-7B-Chat	22.10	31.84	26.97
Starling-LM-7B-beta	14.99	28.70	21.84
zephyr-7b-alpha	25.99	16.43	21.21
vicuna-7b-v1.5-16k	23.14	16.69	19.91
Yi-6B-Chat	18.17	20.05	19.11
zephyr-7b-beta	9.57	22.57	16.07
Llama-2-7b-chat-hf	20.00	11.53	15.77
Qwen1.5-4B-Chat	11.90	16.78	14.34
vicuna-7b-v1.5	16.19	9.87	13.03
Qwen1.5-0.5B-Chat	9.27	10.61	9.94
Qwen1.5-1.8B-Chat	0.98	9.13	5.06

Table 7: **IMO/USAMO results for each hardness level across 34 models.**

Hardness level	IMO	USAMO	Avg.
Easy	54.68	60.27	57.48
Medium	25.79	33.41	29.60
Hard	15.96	22.25	19.11

Subtask	**Subtask Prompt**
Paraphrase	Please paraphrase based on the sentences provided.
Summarize	Please summarize based on the sentences provided.
Simplify	Please explain in simpler terms what this text means.
Story Generation	Please generate a story based on the sentences provided.

Table 8: The subtask prompt for each subtask used in the full prompt.

Table 9: Statistics for tasks in `LiveBench`. This table gives the number of questions for each task, as well as the mean and std. dev number of output tokens per question for `gpt-4-turbo-2024-04-09`.

			Output Tokens	
Category	**Task**	**Num.**	**Mean**	**Std. Dev**
data_analysis	tablejoin	50	78.12	47.55
data_analysis	tablereformat	50	646.18	555.33
instruction_following	summarize	50	223.38	114.28
instruction_following	paraphrase	50	274.78	106.25
instruction_following	story_generation	50	414.94	162.10
instruction_following	simplify	50	229.96	116.67
language	typos	50	215.72	76.93
language	connections	50	462.62	596.09
language	plot_unscrambling	40	626.10	168.73
reasoning	house_traversal	50	154.42	39.18
reasoning	web_of_lies_v2	50	541.50	110.73
reasoning	zebra_puzzle	50	639.56	111.20
math	olympiad	36	765.36	446.36
math	AMPS_Hard	100	581.27	259.08
math	math_competitions	96	676.91	311.95
coding	LCB_generation	50	383.23	285.21
coding	coding_completion	38	129.55	81.39
total		960	412.23	264.63