— 12 —
The problem of aggregation

Multi-task benchmarks promise a holistic evaluation of complex models. An analogy with voting systems reveals limitations in multi-task benchmarks. Greater diversity comes at the cost of greater sensitivity to artifacts.

Source: The Emerging Science of Machine Learning Benchmarks. M. Hardt, 2025. URL: `https://mlbenchmarks.org`. Compiled on 2025-11-03.

Benchmarking is at risk of becoming a victim of its own success. There are hundreds of recent benchmarks and the number is rapidly growing. In the previous chapter we discussed how these benchmarks all give different rankings, at least under direct evaluation. This puts us in a tight spot: Which ranking should we actually look at?

The first option is to punt the problem to the user. Put all the evaluations out there in one big benchmark bazaar and let everyone pick for themselves. The more, the merrier. Although convenient, the idea of a benchmark bazaar runs into two sticky problems. First, it defeats the point of the iron rule. What should model builders and researchers compete over if there are hundreds of incompatible rankings? Second, by what logic should anyone select a model? The benchmark consumer can try to guess what benchmark is most relevant to them, or try out all the different models for themselves somehow. But that is exactly the kind of guesswork benchmarking was supposed to prevent.

Organizing competition and supporting model selection are two vital functions of a healthy benchmarking ecosystem. Both break down when there are too many different benchmarks. It's perhaps for this reason that the machine learning community tends to defy benchmark plurality. At any point in time, the community's attention gravitates toward a few highly influential benchmarks. People prefer to have one number. AI researcher Jason Wei, known for building LLMs at Google, OpenAI, and Meta, sums it up:

> It's critical to have a single-number metric—I can't think of any great evals that don't have a single-number metric.[1]

Optimal decisions compare two alternatives according to a single criterion. We learned this in Chapter 2. A benchmark bazaar simply presents too much information to be useful. It provides practitioners with no guidance about how to select models. It exposes too many competing signals for model builders. And it fails to lay out the rules of a game that the community can compete over.

So, what should we do instead? The only alternative to a bazaar is to somehow reduce the information into a more manageable product. At the extreme end, we can try to aggregate all evaluations into a single representative ranking. That is what multi-task benchmarks try to accomplish. The goal is simple, but the problem isn't. This chapter is about the challenges that arise when trying to aggregate evaluations.

12.1 *Multi-task benchmarks*

Before we make the notion of a multi-task benchmark more precise, let's take a look at some of the most well-known multi-task benchmarks today. Our focus is on how these different benchmarks attempt to solve the aggregation problem.

BIG-Bench. The Beyond the Imitation Game Benchmark (BIG-Bench) came out of a massive collaboration of hundreds of contributors, collecting 214 language modeling tasks. The stated goal of the benchmark was "measuring and extrapolating the capabilities of language models".[2] In doing so, BIG-Bench wasn't the first benchmark that aimed to go beyond the Turing test, originally called *Imitation Game*. Remember the Winograd Schema Challenges from Chapter 10. But BIG-Bench certainly notched up the benchmarking game in terms of task diversity.

The 214 tasks deliberately try to cover all sorts of different challenges, ranging from typical NLP problems to more creative new entries. The task `emoji_movie`, for example, asks to map emoji sequences to movie titles. A rat and two chefs stands for *Ratatouille*. A girl and three fish is *Finding Nemo*. A very different kind of task, `self_awareness`, aims to measure "to what extent the language model is self-aware."[3] Yet another task, called `mnist_ascii` … well, you guessed it:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@   @@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@   @@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@##*@#++#@@@@   @@@@@@@@@@@@@@@@@@@@*+@@@@@@
@@@@@@@@=-:::.....*#**#@@@@@   @@@@@@@@@@@%+====-+-.=@@@@@@
@@@@@@@@%*%:.=@@@#@@@@@@@@@@   @@@@@@@@@@+..*#@@*..=@@@@@@@
@@@@@@@@@@@#::%@@@@@@@@@@@@@   @@@@@@@@@@%-.-%+:.-%@@@@@@@@
@@@@@@@@@@@@@*-::+#@@@@@@@@@   @@@@@@@@@@@+....=#@@@@@@@@@@
@@@@@@@@@@@@@@@@%..:%@@@@@@@   @@@@@@@@@#:..:.-@@@@@@@@@@@@
@@@@@@@@@@@@@#+-:..-@@@@@@@@   @@@@@@@%=.=#@=.=@@@@@@@@@@@@
@@@@@@@##*=:..-=*#@@@@@@@@@@   @@@@@@@+.-#+:-+%@@@@@@@@@@@@
@@@@*:...--+#@@@@@@@@@@@@@@@   @@@@@@@#-.:-#@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@   @@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

The digits are *5* and *8*. The task remains tricky even for recent models.

Many BIG-Bench tasks are eclectic and perhaps somewhat questionable. The community largely avoided the full task plurality of BIG-Bench from the get-go. Various much smaller subsets of BIG-Bench quickly became more widely used. Part of the original release, BIG-Bench Lite (BBL) is a subset of 24 tasks, meant to give "a canonical measure of model performance, while being far cheaper to evaluate."[4] BIG-Bench Hard (BBH) groups 23 of the more challenging tasks from BIG-Bench into a different multi-task benchmark.[5]

It's not just the cost of evaluation. There's a deeper problem with task plurality. Different tasks come with different evaluation metrics. Quantities with different units are *incommensurate* with each other: You can't meaningfully average inches and kilograms, even though both are real numbers. Similarly, there's no coherent meaning to an average of a BLEU-score and a classification accuracy. The quantities might still vary in tandem. Stronger people are often both taller in inches and heaver in kilograms. So, perhaps my "inchgram" number correlates well with *strength*. Similarly, BLUE-scores and accuracy numbers typically both grow with model scale. Averages of incommensurate quantities might still correlate in some intuitive ways with other quantities, but there's no clear meaning to that average.

The incommensurability problem is a bit more subtle still. Two accuracy numbers can also be incommensurate. In binary classification, random guessing gets you 50% accuracy. With many classes random guessing scores close to 0% accuracy. Add varying levels of noise to your problem, and the best possible classifier can come in at any accuracy level, 20%, 50%, or 100%. As a result, it can be a lot easier to make a 10% improvement in some tasks than others, just like it's easier for adults to gain weight than to grow taller.

Multi-task benchmarks, like BIG-Bench, typically take a pragmatic approach to the aggregation problem: Normalize the scores to be in a common interval, say, $[0,1]$, and average them out. BIG-Bench uses a *dynamic range* normalization:

$$\text{normalized score} = 100 \times \frac{\text{raw score} - \text{low score}}{\text{high score} - \text{low score}},$$

based on suitable task-specific choices of *low* score and a *high* score.

OpenLLM leaderboard. Launched by Hugging Face in 2023, the OpenLLM leaderboard attracted participation from 300,000 users and saw more than two million unique visitors in its first ten months.[6] The leaderboard evaluated open weight models through the Eleuther AI evaluation harness, a major open source effort.[7] An updated leaderboard (v2) launched in June 2024, featuring six benchmarks:

- MMLU-Pro, a refined version of MMLU.[8]
- GPQA, difficult "Google-Proof" multiple choice questions about biology, physics, and chemistry.[9]
- MuSR, multistep reasoning featuring algorithmically generated reasoning challenges.[10]
- MATH, a dataset of challenging math problems in a specific format.[11]
- IFEval, an instruction following benchmark.[12]
- BIG-Bench Hard, as discussed above.

In addition to the new benchmarks, v2 changed the scoring. The original leaderboard summed up accuracy numbers across benchmarks. The new leaderboard normalized them to be between the performance of a random baseline and the largest possible score. This effectively decreases the weight of benchmarks where random guessing scores relatively high, e.g., multiple choice questions with few alternatives. The introduction of the new leaderboard caused significant changes to the ranking. Qwen1.5-32B-Chat, for example, moved from rank 57 into the top 10.[6] Interestingly, the top model at the time, Qwen2-72B-Instruct, ranked first in both rankings.

Despite its popularity, Hugging Face shut down the leaderboard in March 2025. A brief announcement cited concerns that benchmarks had evolved and that the leaderboard might become misleading:

> [W]e feel it could encourage people to hill climb irrelevant directions in the field.[13]

Model merging was one of the common tricks to get ahead on the leaderboard. If on different tasks different models scored highest, you'd always get an "easy" improvement by combining the two models. Mixture of Experts (MoE) models route different prompts to different models, hoping to select the best model for the task. In response, Hugging Face created a different category for model merges. Merging isn't cheating though. Rather it points at a fundamental problem with the idea of competing over multiple tasks at the same time.

Holistic Evaluation of Language Models. Holistic Evaluation of Language Models (HELM) is a major effort to evaluate foundation models across many different tasks, so-called *scenarios*.[14] HELM provides several different leaderboards.

The first version of HELM took an innovative approach to aggregation. Instead of averaging out numbers, it computed an overall ranking from win rates. The *win rate* of a model corresponds to the fraction of models that have a lower or equal score on the task. In other words, it's the quantile of the model's score in the task ranking. The *mean win rate* averages this number across all tasks.

What's appealing about mean win rate is that it can be computed from task rankings alone. We don't need to know what metric each task uses, how to normalize it, or how to compare it to other metrics. The whole issue of incommensurability disappears. We simply average out ranking quantiles.

To give a concrete example, consider two models A and B and three tasks. Model B scores higher on two of the tasks, resulting in the mean win rate $\frac{1}{3}(1+1+1/2) = 5/6 \approx 0.83$. Model A has a mean win rate of $\frac{1}{3}(1/2+1/2+1) = 2/3 \approx 0.67$. Model B is better according to mean win rate.

Now add a third, weaker model to the picture. Model C is a smaller variant of model A and therefore scores lower than model A on all tasks. But in that one task where A is better than B, C is also better than B. We get the following win rates:

Win rates	**Task 1**	**Task 2**	**Task 3**	**Mean**
Model A	2/3	2/3	1	7/9
Model B	1	1	1/3	7/9
Model C	1/3	1/3	2/3	4/9

The mean win rate of A is now $\frac{1}{3}(2/3+2/3+1) = 7/9 \approx 0.78$, equal to the mean win rate of B, that is, $\frac{1}{3}(1+1+1/3) = 7/9$. Model C is far off with mean win rate $4/9 \approx 0.44$. What happened here is that introducing a weak model at the bottom of the ranking changed our comparison of the top contenders. Whereas A looked worse than B at first, the two models tie for first place after adding a weaker competitor to the mix.

Mean win rate depends on the set of models under comparison. Rankings can therefore change if you add or remove models, even if these models are in a sense *irrelevant*. Unfortunately, this isn't just a wrinkle of the win rate metric that some clever tweak could iron out. As we'll see later, this is the inevitable property of any aggregation method that works from ranking information alone. In fact, this sensitivity to irrelevant changes was the reason why later versions of HELM moved away from win rate to mean score rankings. The documentation of the 2025 HELM Capabilities leaderboard explains:

Top-level Aggregation. The models are ranked based on the mean

> score, which aggregates metrics across scenarios with the [WildBench] score (1-10) rescaled to 0-1. Note that this is different from our previous approach in HELM Classic and HELM Lite, which is to use the mean win rate as the top-level aggregate score. This change is motivated by the fact that the mean win rate is 1) dependent on the set of models being compared, and 2) sensitive to small variations in scenario scores that invert ranks.[15]

As of October 2, 2025 (release v1.14.0), the top of the HELM Capabilities leaderboard looked like this:

Model	Mean	MMLU-Pro	GPQA	IFEval	WB	Omni-MATH
GPT-5 mini	0.819	0.835	0.756	0.927	0.855	0.722
o4-mini	0.812	0.820	0.735	0.929	0.854	0.720
o3	0.811	0.859	0.753	0.869	0.861	0.714
GPT-5	0.807	0.863	0.791	0.875	0.857	0.647

Curiously, the leaderboard has *GPT-5 mini* slightly ahead of GPT-5, even though "mini" was meant to be a smaller and faster version of GPT-5, not a more capable one. Before GPT-5, OpenAI advertised o3 as its *most capable* model and o4-mini as a smaller, faster alternative. Yet, o4-mini scores about the same as o3. In fact, all four models achieve just about the same mean score. Are they all equally capable or does the mean score fail to measure differences in true capabilities?

Chatbot Arena. Although not a typical multi-task benchmark, Chatbot Arena also faces a massive ranking aggregation problem. We can think of each Arena user as providing a partial model ranking. The platform turns a huge number of such rankings into one central ranking, thus solving an aggregation problem.

In this context, the Elo method from the last chapter is fundamentally a ranking aggregation method. Elo turns many partial rankings into a single ranking via a one-dimensional Bradley-Terry model.

To recall, each model A has a rating R_A, a scalar variable capturing the model's latent skill. When model A and model B face off, Elo computes a probability p that model A wins over model B based on the rating difference $R_A - R_B$. Specifically, Elo assumes that the log odds of model A winning a comparison against model B are proportional to the skill difference:

$$\mathrm{logit}(p) \propto R_A - R_B$$

Elo updates ratings after each matchup so as to improve the fit of the model on the observed comparison. The overall ranking sorts models in descending order of their ratings. Following the mantra of a single number, Chatbot Arena commits to a single ranking in each of a few broad domains, such as text, vision, or multimodal models.

12.2 Problems of aggregation and voting systems

At the outset, most multi-task benchmarks are just a collection of conventional benchmarks, each corresponding to one task. Each task provides a ranking in the usual way. There's a set of test cases and a metric; we rank the models according to their performance on the test cases in terms of the metric choice. You could think of multi-task benchmarks as simple lookup tables for single task performances. From this perspective, there's nothing fundamentally new about multi-task benchmarks.

What makes a multi-task more interesting is the idea of a single representative ranking that tries to capture overall performance across all tasks. Abstractly, the multi-task *ranking aggregation problem* is the following:

- **Input**
  - A set $\mathcal{M}$ of models $A, B, C, D, \ldots$
  - For each task i, a partial ordering $A \succ_i B$ indicating that task i ranks model A higher than model B
- **Output**
  - A complete ordering $A \succ B$ over all models.

This aggregation problem applies to the original HELM benchmark, which computes an aggregate ranking from individual task rankings via the mean win rate. It also applies to Chatbot Arena if we think of each visitor as one task giving us one comparison. But the problem doesn't apply to benchmarks that consider numerical scores, like BIG-Bench or the OpenLLM leaderboard. Therefore, we define the multi-task *rating aggregation problem* analogously:

- **Input**
  - A set $\mathcal{M}$ of models $A, B, C, D, \ldots$
  - For each task i and model A a rating $r_i(A)$
- **Output**
  - A complete ordering $A \succ B$ over all models.

Rating aggregation allows each task to specify a number for each model. The aggregation rule takes these numbers into account.

Voting. Both aggregation problems have an equivalent interpretation in terms of *voting*. Think of tasks as voters and models as candidates. A voting system has to aggregate the votes of each voter into an outcome, such as a single winner a full ranking of all candidates. Therefore, multi-task benchmarks are analogous to voting systems where tasks are voters and models are candidates.

In the context of voting, multi-task ranking aggregation is called *ranked voting*, while *rated voting* corresponds to multi-task rating aggregation. Voting has a rich history in political theory, economics, and the social sciences more broadly. In particular, voting is a central subject of the broader field of *social choice theory*.

Social choice theory studies how groups of individuals come to agree on a social outcome. The outcome could be the winner of an election, how to divide up a shared resource, or the choice of a policy in a city government. The focus of social choice theory is on collective decision-making processes.

The connection to social choice theory has two useful applications. First, it illuminates the kind of problems we're going to encounter with multi-task benchmarks. Voting has a rich theory that exposes fundamental challenges with aggregation. Second, social choice theory has developed numerous clever voting rules, many of which have never been implemented in benchmarking. It's entirely possible that future multi-task benchmarks will implement clever ideas from the theory of voting.

To summarize the connection with voting:

- Tasks are voters.
- Models are candidates.
- Multi-task benchmarks are voting systems.

Ranked voting systems are also called *ordinal voting systems.* Ordinal systems only take the relative preference of voters into account. All that matters is whether a voter prefers candidate A to candidate B or the other way around. Rated voting, on the other hand, corresponds to *cardinal voting systems*. Cardinal systems take the strengths of votes into account. Votes numerically represent the strength of preference for a candidate.

Both ordinal and cardinal voting systems make sense for benchmarking and have been implemented in major multi-task benchmarks. We'll go over ranked voting and rated voting in turn, covering some of the insights from the social choice literature relevant to benchmarking. Perhaps the single biggest lesson from social choice theory is that there's no voting system that checks all the boxes. All voting systems fail to have some desirable property in some cases at least. This sounds disappointing, but social choice theory has also developed many ways to cope.

12.3 *Ranked voting systems*

Ordinal voting rules are common in political elections and have long been studied in social choice theory. Recall our formal setup:

- There are *candidates* or *alternatives* $A, B, C, D, \dots$.
- Voter i has preferences $A \succ_i B$, meaning that the voter prefers candidate A over candidate B.

In addition, we make two assumptions about every voter i:

- **Completeness**: For any two alternatives A, B, either $A \succ_i B$ or $B \succ_i A$.
- **Transitivity**: $A \succ_i B$ and $B \succ_i C$ implies $A \succ_i C$.

These assumptions are also natural in the machine learning context, because they are equivalent to having a ranking of all models for each task.

Fact 1. *A complete and transitive preference relation is equivalent to a ranking.*

Proof. It's not hard to see that any ranking gives us a complete and transitive preference relation. But let's see how to go from a complete and transitive preference relation to a ranking. The idea is to repeatedly "peel off" the top candidate until the ranking is complete. What's not obvious is that a "top" candidate is always well-defined.

Given a preference relation, consider the candidate A that beats *the most* other candidates in pairwise comparisons. Such a candidate must always exist. It turns out that A must, in fact, beat all other candidates. Arguing by contradiction, suppose there were a candidate B that beats A. Then, by transitivity, B beats everything A beats. Then, by completeness, these are all pairwise comparisons. Hence, B beats more alternatives than A in pairwise comparisons.

The previous observation is the key argument. Using this claim, we can construct a ranking by repeatedly "peeling off" the top candidate that beats the most other alternatives until the ranking is complete.

□

Majority vote and Condorcet's paradox

There's a natural voting rule for two alternatives A, B. It's the majority vote. Take the alternative that is preferred by most voters. Chapter 9 covered some of the nice properties of majority voting in the context of label aggregation.

Can we generalize the majority vote to more than two alternatives? One natural generalization of a majority winner to more than two alternatives is the notion of a *Condorcet winner*.

Definition 1. *A* Condorcet winner *is a candidate that would get more than half the votes in a one-on-one race against any opponent.*

Attempting to find a Condorcet winner, we can define the aggregate preference relation $A \succ B$, if $A \succ_i B$ for a majority of voters. This relation is complete, assuming all voters have complete preferences. If the aggregate preference relation is also transitive, we have a ranking and the Condorcet winner is at the top of this ranking. Unfortunately, the aggregate relation need not be transitive. It can, in fact, have cycles. This stumbling block is known as *Condorcet's paradox.*

Condorcet's paradox. A minimal example of Condorcet's paradox has three candidates A, B, C and three voters with the following preferences:

- Voter 1: $A \succ_1 B \succ_1 C$
- Voter 2: $B \succ_2 C \succ_2 A$
- Voter 3: $C \succ_3 A \succ_3 B$
- Aggregate preference relation:
  - $A \succ B$ (voter 1, 3)
  - $B \succ C$ (voter 1, 2)
  - $C \succ A$ (voter 2, 3)

Clearly, the aggregate preference relation is cyclic! As a result, there is no Condorcet winner.

Fact 2. *Complete and transitive voter preferences may not have a Condorcet winner.*

Condorcet's paradox also says something about tournaments. In a tournament, candidates repeatedly face off in pairs until a single winner remains. Condorcet's paradox implies that the winner depends on the order of play. This incentivizes candidates to be strategic about the order in which they're being compared.

Condorcet method. Earlier we defined majority relation so that $A \succ B$ if a majority of voters prefer A over B. If this relation is transitive, there is a Condorcet winner: Compute the ranking and pick the top. But if it isn't transitive, we're stranded. In particular, this approach doesn't define a voting system.

A *Condorcet method* is any valid voting system that always finds a Condorcet winner, if it exists. A simple Condorcet method is *Ranked Pairs* voting. Ranked Pairs find an acyclic subset of the majority relation, while prioritizing stronger victories:

1. Compute the winner for each pairwise matchup between two candidates.
2. Sort those victories by margin of victory from strongest to weakest.
3. Create a directed acyclic graph where each node is a candidate. For each matchup in sorted order, add an edge from winner to loser, if it doesn't create a cycle. Skip any matchup that would create a cycle.
4. The source of the graph is the Condorcet winner.

In fact, the directed acyclic graph gives a complete and transitive relation that gives a full ranking of candidates.

Positional voting

Positional voting is a ranked voting system where candidates get points for their position in each of the voters' rankings. There are many different ways to assign points based on ranks. One well-known example is the *Borda count*.

According to the Borda count, the top candidate in a ranking of k candidates gets $k-1$ points, the second gets $k-2$, and so on. The last candidate in the ranking gets 0 points. We add up all the points for each candidate and sort candidates by their overall point score in descending order.

Up to scaling, Borda count is equivalent to mean win rate. The win rate assigned 1 to the top candidate, $1-1/k$ to the second, $1-2/k$ to the third and so on. Multiply the win rate by k and subtract 1 to get the Borda count.The two quantities have a linear monotone relationship.

Compared with Condorcet's method, positional voting directly produce a ranking, i.e., a complete, transitive aggregate preference. There's always a winner and we don't need to worry about cycles. The problem with positional voting is that the ranking of the top contenders depends on how they perform against far off candidates, so-called *irrelevant alternatives*.

We already saw one such example earlier on in the context of HELM. Here's another example in the language of voting to illustrate the same point. Suppose 90 voters prefer a candidate A and only 10 voters prefer the alternative B. The Borda count of A is 90 and the Borda count of B is 10. Candidate A is the clear winner. But now introduce another 98 candidates so that there are 100 candidates and 100 voters. Let's say 90 voters still have A first and B second. But those 10 voters that have B first really dislike A and rank it last among the 100 alternatives. We can compute the counts:

- Borda count for A equals $90 \times 99 = 8910$
- Borda count for B equals $90 \times 98 + 10 \times 99 = 9810$
- The Borda count for any other candidate is at most $10 \times 98 = 980$.

The example shows how weak candidates with low Borda counts can flip the ranking of the top contenders.

Arrow's theorem

Given the issues with Condorcet's method and positional voting, you might ask if there any ranked voting systems for many candidates that have all the desirable properties? Arrow's impossibility result from 1950 suggests a negative answer to this question. We must always give up on something that we'd like to have. To make this point, Arrow puts forward two criteria for a good voting system:

- **Unanimity**: If every voter prefers A over B, then A should rank higher than B.
- **Independence of irrelevant alternatives (IIA)**: The ordering of A relative to B should only depend on how each voter compares A to B.

In addition, assume that the voting system should not restrict voters: Each voter may rank candidates any way they want. For two alternatives, majority voting satisfies both unanimity and IIA. However, Arrow's theorem says that there is no such voting rule for three or more alternatives.[16,17]

Theorem 1. *If there are at least three alternatives, then any voting system that satisfies both unanimity and IIA must be a dictatorship by one voter.*

A dictatorship by one voter corresponds to the case where the aggregate ranking is identical to the ranking of one single voter, the "dictator". Arrow's theorem has an intuitive interpretation in terms of benchmarking. A multi-task benchmark that checks all the boxes... *isn't*: actually, it must be a single-task benchmark in disguise. To get something that's truly not a single-task benchmark, we'll have to give up on at least one of the two properties, unanimity or IIA.

Having said that, Arrow's theorem is a theoretical result that applies to ranked voting, but not to rated voting. This opens up the possibility that perhaps rated voting can avoid all issues. Moreover, even for ordinal voting systems, perhaps Arrow's theorem doesn't actually occur in practice. We'll discuss each of these possibilities in turn.

12.4 Rated voting

In rated voting, voters express the strength of their votes, not just the ordering of different candidates. *Score voting* is one of the most common rated voting mechanisms. In score voting, voters express their preference for each candidate on some numerical scale. The candidate with the highest average score wins. Score voting is also the most common variant of rated voting in the context of benchmarking. Social choice theory has considered many other rated voting systems and hybrids between ranked and rated voting. Here, we'll focus on score voting. Our first observation is that like any other voting method, score voting is no panacea.

Fact 3. *Score voting may not elect a Condorcet winner.*

A simple example demonstrates the fact.

	Voter 1	**Voter 2**	**Voter 3**	**Mean**
A	0.8	0.8	0.2	0.6
B	0.7	0.6	0.8	**0.7**
C	0.4	0.1	0.4	0.3

Model B wins the score vote, but model A is the Condorcet winner: It wins on 2 out of 3 tasks against any other model. Mapping this example to benchmarking, do you prefer model A or model B?

It depends. Model A largely loses the score vote, because it scores low in Task 3 (Voter 3). Depending on the benchmark, it's possible that Task 3 is a quirky task, say, `mnist_ascii`, where model A just fails. The company behind model B may have recognized the importance of Task 3 for the overall ranking and specifically prepared their model for this task (cf. Chapter 11). This might explain the unusually high score of model B on Task 3. In this case, we're better off with the Condorcet winner. In an alternative scenario, however, Task 3 might point at an important shortcoming of model A. In this case, we're better off trusting the score vote.

Strategic behavior

Rated voting avoids Arrow's theorem, but it nevertheless runs into a similar predicament. Any ranked voting system with more than three alternatives is either dictatorial or it incentivizes voters to act *strategically*: To get the outcome they desire, voters will generally want to report preferences as a function of what they believe other voters do. In particular, voters will misrepresent their true preferences. This predicament is a consequence of Gibbard's 1973 theorem that has several variants and holds in quite some generality.[18,19]

There are many ways to vote strategically. You might pick the "lesser evil" instead of your preferred candidate, because the lesser evil has the better chance to win. You might state overly low preferences for your second choice to make it more likely that your preferred choice wins.

What does strategic behavior mean for benchmarking? The number of tasks in a benchmark is relatively small compared to the number of voters in many elections. Generally, strategic behavior is more potent when the number of voters is small. On the other hand, voters in benchmarking are tasks. It's unlikely—though not impossible to imagine—that benchmark creators strategically design a task so as to benefit a specific model. What's more likely, however, is that model builders strategically invest effort towards improvements in some tasks versus others. Strategic effort allocation has a similar effect on outcomes as strategic voting.

Strategic effort allocation in multi-task benchmarks. We can learn a lot from a simple setting of *strategic effort allocation*. In this simple setup, a model builder invests effort $x_i \geq 0$ into preparing the model for task i. We assume that performance improves with effort as a non-negative function $f_i(x_i) \geq 0$ that is differentiable and strictly concave. That is, there are diminishing returns to effort, as is usually the case. The model builder will spend one unit of effort across all k tasks so that $\sum_{i=1}^{k} x_i = 1$.

The goal of the model builder is to maximize the total score improvement. Assuming the benchmark averages out scores across tasks, the model builder therefore wants to maximize the sum of score improvements across all tasks.

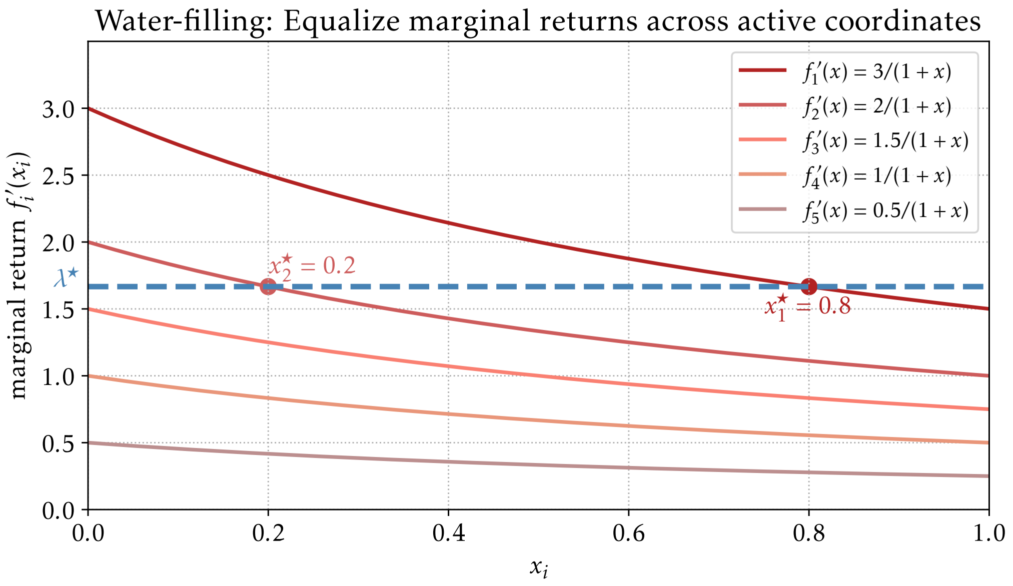

Figure 12.1: Optimal strategic behavior in a multi-task benchmark with five tasks via water filling: Lower the water level (dashed line) from top to bottom until you find coordinates of equal marginal return that sum up to 1.

This corresponds to the optimization problem:

$$\max_{x_i \geq 0} \quad \sum_{i=1}^{k} f_i(x_i) \quad \text{s.t.} \quad \sum_{i=1}^{k} x_i = 1$$

The problem has a nice optimal solution, illustrated by a "water filling" argument. The water level corresponds to the current marginal return that we achieve. We start high and continually decrease the water level until we fully spend our budget.

More precisely, start from the coordinate i_1 with the highest marginal return $f_{i_1}'(0)$ at 0. As you increase this coordinate x_{i_1}, the marginal return $f_{i_1}'(x_{i_1})$ decreases. Eventually, you hit a second coordinate i_2 whose marginal return $f_{i_2}'(0)$ at 0 is equal to $f_{i_1}'(x_{i_1})$. Now you increase both of these coordinates while equalizing their marginal returns. Eventually, you activate a third coordinate, and so on. Continue until you deplete the budget of effort, that is, the active coordinates sum to 1. At this point, you will have found an optimal marginal return $\lambda^\star \geq 0$ that's equal across all active coordinates.

The optimal effort allocation prioritizes tasks with the greatest marginal returns. Moreover, the optimal solution is typically sparse. Most coordinates are 0 and only a few are positive. This means that cardinal multi-task benchmarks generally incentivize model builders to invest effort in a polarized manner, concentrating on a few influential tasks. Moreover, these tasks are the "easy" tasks, where improvements are relatively cheap to come by.

In a nutshell, to strategize in a multi-task benchmark, identify easy tasks with large marginal returns to effort. Concentrate your efforts on those tasks and neglect the other tasks. It's easy to imagine, however, that this *strategic optimum* doesn't point toward the best way to actually improve a model. Then again, over time competition might lead to an equilibrium where all tasks have the same marginal returns. The reason is that if repeatedly applying the water filling strategy eventually converges to uniform marginal returns.

From the perspective of the benchmark designer, it makes sense to anticipate strategic behavior by picking tasks with equal marginal returns to effort. This will incentivize competitors to invest in broad improvements across all tasks.

12.5 *Empirical trade-offs in multi-task benchmarks*

Do any of the theoretical issues we've encountered actually matter in practice? Perhaps these are just contrived counterexamples that don't come up in real multi-task benchmarks. The theory also didn't give us any quantitative sense about how bad the problem is. Yes, ranked voting is sensitive to irrelevant alternatives, but how sensitive is it? Likewise, rated voting is sensitive to rescaling of preferences, but how much can that actually affect rankings? We'll now see an empirical answer to both questions.

As it turns out, there is something like an empirical analogue of Arrow's theorem for multi-task benchmarks. It applies to both cardinal and ordinal aggregation. Loosely speaking, any multi-task benchmark that's far from a single task must be sensitive to irrelevant changes. So, if the benchmark is truly multi-task, it must pay a price. To make the statement more precise, we need to say what counts as irrelevant changes and what it means to be far from a single task.

Measuring sensitivity to irrelevant changes in benchmarks

In line with social choice theory we put forward desirable properties of multi-task benchmarks. We'll then show that the only multi-task benchmarks that satisfy them are essentially single task benchmarks.

- **Independence of irrelevant models:** Adding low-performing models to a multi-task benchmark shouldn't change the order of top contenders.
- **Independence of irrelevant task transformations:** Changing a task metric from s to $as + b$ for $a > 0$ shouldn't change the overall ranking.

The first property seems like a no-brainer from a benchmarking perspective. It shouldn't be that adding a low-performing model to the benchmark changes the order of the top two contenders. This criterion is directly analogous to Arrow's independence of irrelevant alternatives. But it's perhaps even more desirable in a benchmarking context. It's a lot easier to add a model to a leaderboard than it is to run for president. If it's possible to affect the ranking significantly via low-performing model addition, we'll likely see strategic behavior along those lines.

The second criterion asks that equivalent task metrics shouldn't affect the over all ranking. For accuracy benchmarks this criterion has another natural interpretation in terms of label errors. Suppose we add a $p > 0$ fraction of random label errors to a classification benchmark with C classes. Any accuracy number s will change to $(1-p)s + p/C$. We get accuracy s on the original labels and random guessing accuracy $1/C$ on the random labels. But the accuracy transformation is strictly monotone for any $p < 1$. That is, both benchmarks give us identical rankings. The two benchmarks are operationally equivalent. The accuracy numbers have a monotone linear relationship. Chapter 7 covered the empirical fact that accuracy numbers on different benchmarks, like ImageNet and ImageNet V2, often exhibit a monotone linear relationship. Another way to think of the second criterion is that it shouldn't matter whether you put ImageNet or ImageNet V2 in our multi-task benchmark.

The failure of either criterion indicates a sensitivity of the aggregate ranking to changes that shouldn't matter. To measure the degree of sensitivity to irrelevant changes, we need to quantify the magnitude of the change in ranking. Here, Kendall's τ comes in handy again. As in Chapter 11, Kendall's τ measures the ordinal association of two rankings. Recall, assuming no ties, Kendall's τ equals

$$\tau = \frac{C - D}{\binom{n}{2}},$$

where C is the number of concordant pairs and D is the number of discordant pairs. Two perfectly aligned rankings have $\tau = 1$, reverse rankings have $\tau =$

−1. For random rankings, expect $\tau = 0$.

We define *sensitivity* in terms of the quantity

$$\frac{1-\tau}{2} = \frac{D}{\binom{n}{2}} \in [0, 1].$$

With this measure at hand, we can empirically evaluate how large we can make the disagreement between the original ranking and the new ranking under (a) addition of irrelevant models, and (b) irrelevant task transformations.

We can always avoid all problems with aggregation by going to a single-task benchmark. Both properties hold, because there's only one ranking. This is the dictator clause in social choice theory. Arrow's and Gibbard's theorem only apply to voting systems that aren't dictatorial. Likewise, we're not going to say anything about single-task benchmarks. But we have to be careful. We don't get a multi-task benchmark simply by copying the same task many times, or by including many very similar tasks. That would just be a single-task benchmark in disguise. We therefore also need a robust measure of *task diversity* that measures distance from single-task.

Trade-offs between sensitivity and diversity

To quantity the distance of a multi-task benchmark from being single-task, we introduce a measure of *task diversity*. Diversity measures the degree to which different tasks agree in their model rankings. Formally, we utilize Kendall's coefficient of concordance that is often used as a measure of *inter-rater* agreement.

Suppose there are k tasks ranking m models. Denote by r_{ij} the rank assigned by task i to model j. The total rank for candidate j is

$$R_j = \sum_{i=1}^{k} r_{ij},$$

and the mean of total ranks is $\bar{R} = \frac{1}{2}k(m+1)$. *Kendall's coefficient of concordance* is

$$W = \frac{12S}{k^2(m^3 - m)} \quad \text{with} \quad S = \sum_{j=1}^{m} (R_j - \bar{R})^2.$$

The coefficient satisfies $0 \le W \le 1$, where $W = 1$ indicates perfect agreement among the tasks and $W = 0$ indicates no agreement. Random rankings have expected agreement $1/k$.

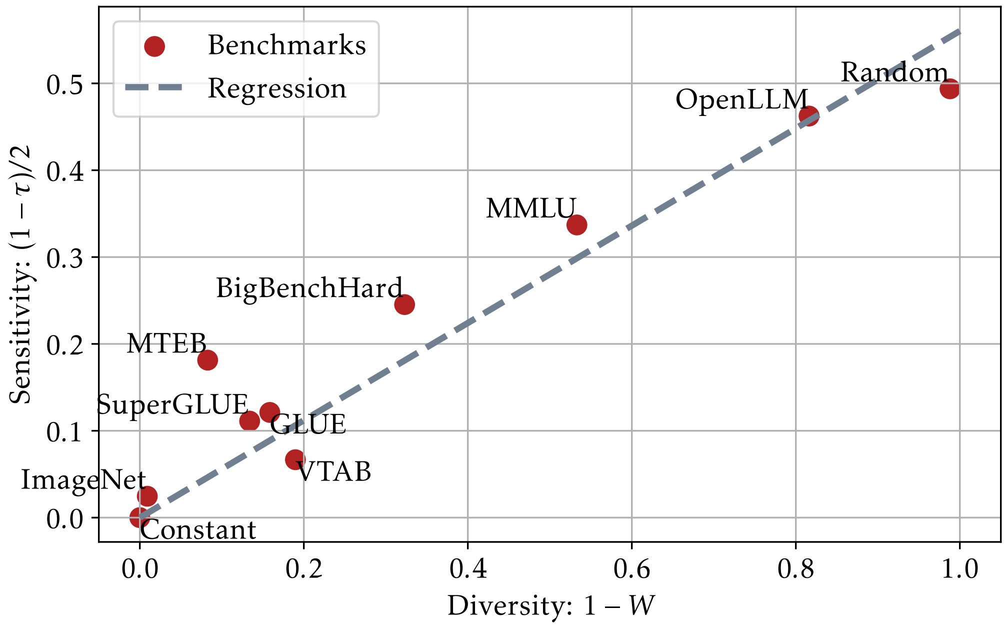

Figure 12.2: Trade-off between sensitivity and diversity for cardinal multi-task benchmarks. Kendall's τ rescaled to $[0,1]$ so that $1/2$ corresponds to random rankings.

To get a measure of diversity, we consider $1-W$. In the context of benchmarking, we interpret Kendall's W as a measure of closeness to single-task. The case $W=0$ corresponds to a single task benchmark, whereas $W=1$ is maximum diversity.

The figure contrasts diversity and sensitivity for a number of well-known multi-task benchmarks. All of them fall on a line between a single-task ranking, constant under irrelevant changes, and a random ranking. What this means is that as you increase diversity, empirically, you also increase sensitivity to irrelevant changes. This isn't a theorem. It's an empirical observation that appears to hold robustly for numerous multi-task benchmarks. There is a similar trade-off for ordinal benchmarks.

12.6 *Latent factors in benchmark performance*

All aggregation rules we've seen so far are fixed, data-independent functions of the individual task information. In principle, nothing stops us from considering data-dependent aggregation rules. One natural idea is to look at the

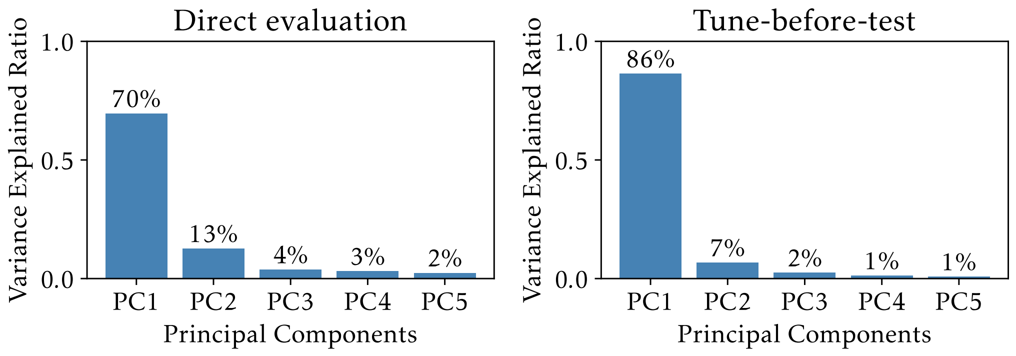

Figure 12.3: Principal components of the benchmark-model score matrix. Left: Scores from direct evaluation. Right: Scores after tune-before-test.

top principal component of the benchmark-model score matrix. Specifically, consider a matrix M such that entry M_{ij} corresponds to the score of model j on task i.

The benchmark-model matrix is close to low rank. If we compute benchmark scores under direct evaluation, we can approximate the matrix well using a rank-5 matrix. In line with Chapter 11, the benchmark-model matrix is even closer to rank 1 when we look at scores under tune-before-test evaluation. Here, the matrix is close to rank 1. In other words, benchmark results are essentially one-dimensional. What is that one dimension?

The first principal component (PC1) corresponds to the vector with the largest singular value of the benchmark-model matrix. This is the direction along which models (row vector in the matrix) have the highest variance. We can compute the projection of each model vector onto the first component (PC1 score). The length of this projection corresponds strongly with the pre-training FLOP count for the model.

This suggests, perhaps intuitively, that benchmark performance is primarily a function of pre-training compute. But it's less clear if there's a strong meaning to the other components. Especially under direct evaluation, the remaining principal components still have non-negligible weight. Researchers have attempted to map these components to specific capabilities that the model might have. It's also plausible that these components have something to do with task adaptation. Since the magnitude of the second principal component shrinks under tune-before-test evaluation, it might tell us how much a model can benefit from task adaptation.

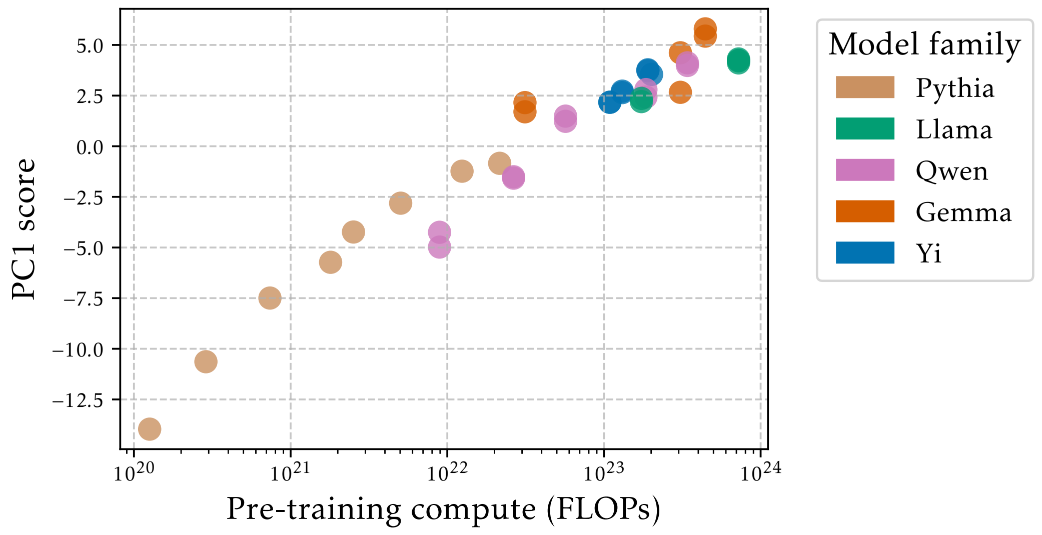

Figure 12.4: The first principal component of the benchmark-model matrix correlates strongly with pretraining compute.

Notes

Social choice theory. Our exposition of voting and social choice draws on an excellent introductory chapter by Easley and Kleinberg.[20] Taylor's text *Social Choice and the Mathematics of Manipulation* is comprehensive reference for social choice theory.[21] We only scratched the surface of the topic. Wikipedia lists 23 single-winner voting systems judged across 17 different criteria. There's a lot more to learn.

Rofin et al. (2023) use social choice theory to design multi-task benchmarks. Specifically, they propose a framework, Vote'n'Rank, combining eight aggregation procedures. With the goal to improve over mean score aggregation, the work evaluates various properties of these alternative aggregation rules.

The trade-offs between diversity and stability in multitask benchmarks are due to Zhang and Hardt.[22] Boehmer et al. find that real-world election outcomes are sensitive to noise, too.[23]

Perspectives on multi-task benchmarks. Bommasani et al. discuss challenges with "meta-benchmarks", what we call multi-task benchmarks.[24] They argue that meta-benchmarks may make it difficult to distinguish genuine improvements from clever ways to adapt to the different tasks. In addition, specific improvements relating to data selection, training objectives, and architectures, may be more difficult to detect from broad meta-benchmarks. McIntosh et al.[25] broadly examine pitfalls with NLP benchmarks, primarily multi-task benchmarks, along numerous criteria. Ethayarajh and Jurafsky apply microeconomic principles to NLP benchmarks to argue that they may poorly capture the utility of the NLP researcher.[26]

Wang, Herzmann, and Russakovsky argue for comprehensive *benchmark suites* to replace leaderboards.[27]

Madaan et al. measure variance in LLM benchmarking results.[28] Dehgani et al. show how different subsets of tasks in multi-task benchmarks can lead to different winners, a problem they call *benchmark lottery*.[29] Zheng et al.[30] show how adversarially created *constant predictors* that always return the same fixed value can score high on multi-task benchmarks. Pacchiardi et al. demonstrate how simple *n*-gram models can solve various BIG-Bench tasks without achieving the capabilities that the task aims to test.[31]

Skill-Mix is a creative take on multi-task benchmarking.[32] Rather than aggregating over tasks, Skill-Mix will mix challenges from different tasks into a single prompt.

Principal components and observational scaling laws. The observation about the low-rank nature of the benchmark-model matrix is from a paper on observational scaling laws.[33] Observational scaling laws are scaling laws for downstream benchmark performance derived from the benchmark-model matrix. The plots here are from Zhang, Dominguez-Olmedo and Hardt, who compared the low-rank approximations of the benchmark-model matrix before and after applying tune-before test (cf. Chapter 11).[34]

Efficient evaluation. One major problem with multi-task benchmarks is that evaluation gets very slow. Even major companies, like Anthropic, found BIG-Bench too "unwieldy" in practice:

> While it was a useful exercise to try to implement BIG-Bench, we found it sufficiently unwieldy that we dropped it after this experiment.[35]

This has motivated much work on methods that speed up evaluation. One approach is to predict performance across many benchmarks from fewer evaluations.[36,37] This is in some sense an alternative to multi-task benchmarking. Rather than aggregating many evaluations, efficient evaluation tries to work from fewer evaluations to begin with. There is, however, evidence that efficient evaluation—while accurate on typical models—may fail to give accurate estimates on new models.[38]

Bibliography

1. Wei, J. *Successful language model evals* https://www.jasonwei.net/blog/evals. Blog post, May 24, 2024. 2024 (↑ 2).
2. Srivastava, A. *et al.* Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research* (2023) (↑ 3).
3. BIG-bench authors. *BIG-bench: Self Awareness task* https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/self_awareness. Accessed: 2025-10-06. 2025 (↑ 3).
4. BIG-Bench authors. *BIG-bench: Beyond the Imitation Game Benchmark* https://github.com/google/BIG-bench. Accessed: 2025-10-06. 2025 (↑ 4).
5. Suzgun, M. *et al.* Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv:2210.09261* (2022) (↑ 4).
6. Fourrier, C., Habib, N., Lozovskaya, A., Szafer, K. & Wolf, T. *Open LLM performances are plateauing, let's make the leaderboard steep again* https://huggingface.co/spaces/open-llm-leaderboard/blog. Hugging Face Spaces blog, retrieved 2025-10-07. 2024 (↑ 4, 5).
7. Gao, L. *et al. A framework for few-shot language model evaluation* version v0.4.3. July 2024. https://zenodo.org/records/12608602 (↑ 4).
8. Wang, Y. *et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark* in *NeurIPS Datasets and Benchmarks Track* (2024) (↑ 5).
9. Rein, D. *et al. GPQA: A Graduate-Level Google-Proof Q&A Benchmark* in *Proc. Conference on Learning on the Move (COLM) 2024* arXiv:2311.12022 (2024) (↑ 5).
10. Sprague, Z., Ye, X., Bostrom, K., Chaudhuri, S. & Durrett, G. MuSR: Testing the Limits of Chain-of-Thought with Multistep Soft Reasoning. *arXiv:2310.16049.* Submitted to openreview / under review (2023) (↑ 5).
11. Hendrycks, D. *et al. Measuring Mathematical Problem Solving with the MATH Dataset* in *NeurIPS Datasets and Benchmarks Track* Also arXiv:2103.03874 (2021) (↑ 5).
12. Zhou, J. *et al.* Instruction-following evaluation for large language models. *arXiv:2311.07911* (2023) (↑ 5).
13. Fourrier, C. *It's been a wild ride, folks :) (End of the Open LLM Leaderboard)* https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard/discussions/1135. Hugging Face Spaces discussion, pinned on Mar 13, 2025. 2025 (↑ 5).
14. Liang, P. *et al.* Holistic evaluation of language models. *arXiv:2211.09110* (2022) (↑ 5).

15. Xu, J., Mai, Y. & Liang, P. *HELM Capabilities: Evaluating LMs Capability by Capability* `https://crfm.stanford.edu/2025/03/20/helm-capabilities.html`. Accessed: 2025-10-06. 2025 (↑ 7).
16. Arrow, K. J. A difficulty in the concept of social welfare. *Journal of political economy* **58,** 328–346 (1950) (↑ 13).
17. Arrow, K. J. *Social choice and individual values* (Yale University Press, 2012) (↑ 13).
18. Gibbard, A. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society*, 587–601 (1973) (↑ 15).
19. Gibbard, A. Manipulation of schemes that mix voting with chance. *Econometrica: Journal of the Econometric Society*, 665–681 (1977) (↑ 15).
20. Easley, D. & Kleinberg, J. *Networks, crowds, and markets: Reasoning about a highly connected world* (Cambridge university Press, 2010) (↑ 22).
21. Taylor, A. D. *Social choice and the mathematics of manipulation* (Cambridge University Press, 2005) (↑ 22).
22. Zhang, G. & Hardt, M. Inherent trade-offs between diversity and stability in multi-task benchmarks. *arXiv:2405.01719* (2024) (↑ 22).
23. Boehmer, N., Bredereck, R., Faliszewski, P. & Niedermeier, R. *A quantitative and qualitative analysis of the robustness of (real-world) election winners* in *ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization (EAAMO)* (2022), 1–10 (↑ 22).
24. Bommasani, R. *et al.* The foundation model transparency index. *arXiv:2310.12941* (2023) (↑ 22).
25. McIntosh, T. R. *et al.* Inadequacies of large language model benchmarks in the era of generative artificial intelligence. *arXiv:2402.09880* (2024) (↑ 23).
26. Ethayarajh, K. & Jurafsky, D. Utility is in the eye of the user: A critique of NLP leaderboards. *arXiv:2009.13888* (2020) (↑ 23).
27. Wang, A., Hertzmann, A. & Russakovsky, O. Benchmark suites instead of leaderboards for evaluating AI fairness. *Patterns* **5** (2024) (↑ 23).
28. Madaan, L. *et al.* Quantifying variance in evaluation benchmarks, 2024. *URL https://arxiv.org/abs/2406.10229* (↑ 23).
29. Dehghani, M. *et al.* The benchmark lottery. *arXiv:2107.07002* (2021) (↑ 23).
30. Zheng, X. *et al.* Cheating automatic llm benchmarks: Null models achieve high win rates. *arXiv:2410.07137* (2024) (↑ 23).
31. Pacchiardi, L., Tesic, M., Cheke, L. G. & Hernández-Orallo, J. Leaving the barn door open for Clever Hans: Simple features predict LLM benchmark answers. *arXiv:2410.11672* (2024) (↑ 23).
32. Yu, D. *et al.* Skill-Mix: A flexible and expandable family of evaluations for AI models. *arXiv:2310.17567* (2023) (↑ 23).
33. Ruan, Y., Maddison, C. J. & Hashimoto, T. B. *Observational scaling laws and the predictability of langauge model performance* in *Neural Information Processing Systems (NeurIPS)* (2024), 15841–15892 (↑ 23).
34. Zhang, G., Dominguez-Olmedo, R. & Hardt, M. Train-before-Test Harmonizes Language Model Rankings. *arXiv:2507.05195* (2025) (↑ 23).
35. Anthropic. *Challenges in evaluating AI systems* Accessed 2025-07-22. `https://www.anthropic.com/news/evaluating-ai-systems` (↑ 23).
36. Vivek, R., Ethayarajh, K., Yang, D. & Kiela, D. Anchor points: Benchmarking models with much fewer examples. *arXiv:2309.08638* (2023) (↑ 23).

37. Polo, F. M. *et al.* tinyBenchmarks: evaluating LLMs with fewer examples. *arXiv:2402.14992* (2024) (↑ 23).
38. Zhang, G., Dorner, F. E. & Hardt, M. How Benchmark Prediction from Fewer Data Misses the Mark. *arXiv:2506.07673* (2025) (↑ 24).