Why LLM Benchmarking is Broken and How to Fix It

Guanhua Zhang

Social Foundations of Computation

Ranking Is All You Need

At the core of applied machine learning are *model rankings*

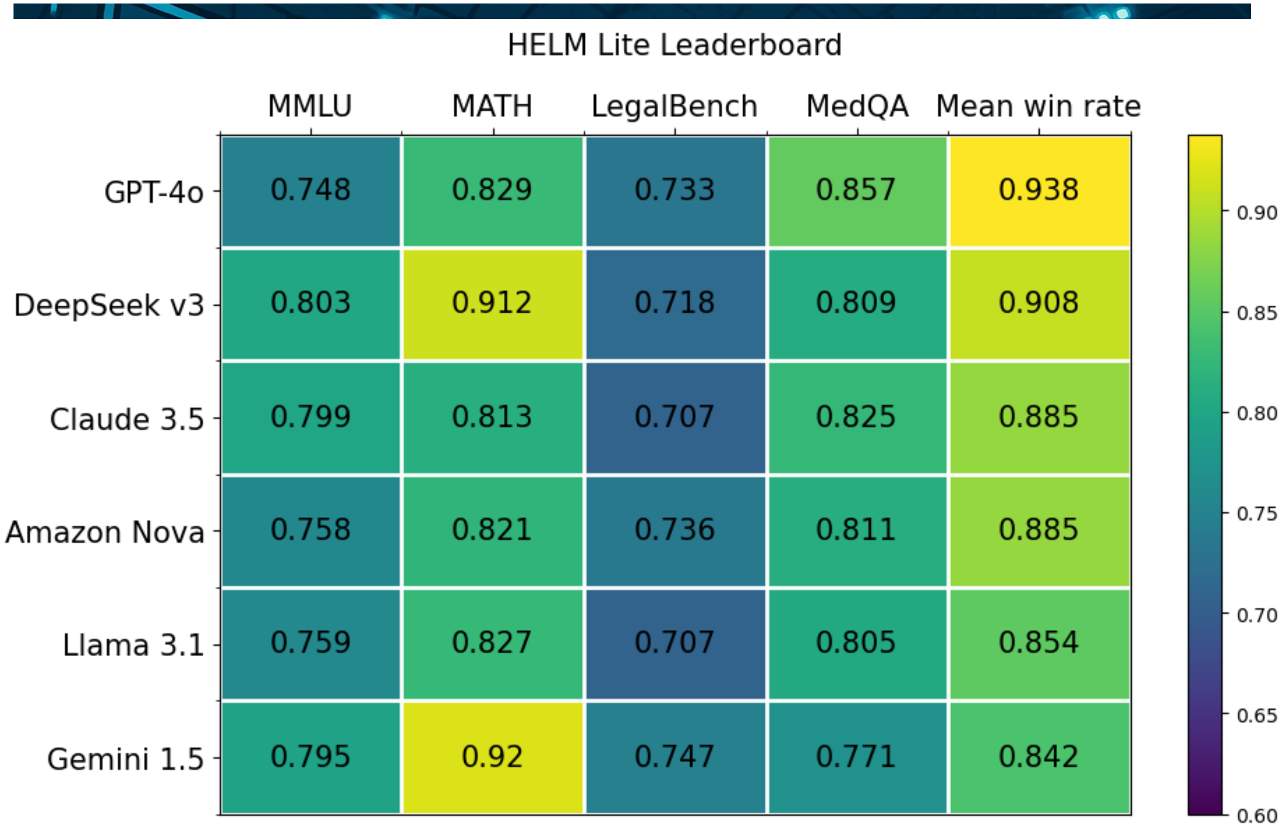
Good model rankings are the goal of benchmarking

Multi-Task Benchmarking for LLMs

LLMs can solve many tasks

Which ranking should we look at?

Multi-task benchmarks: Just evaluate them on everything!

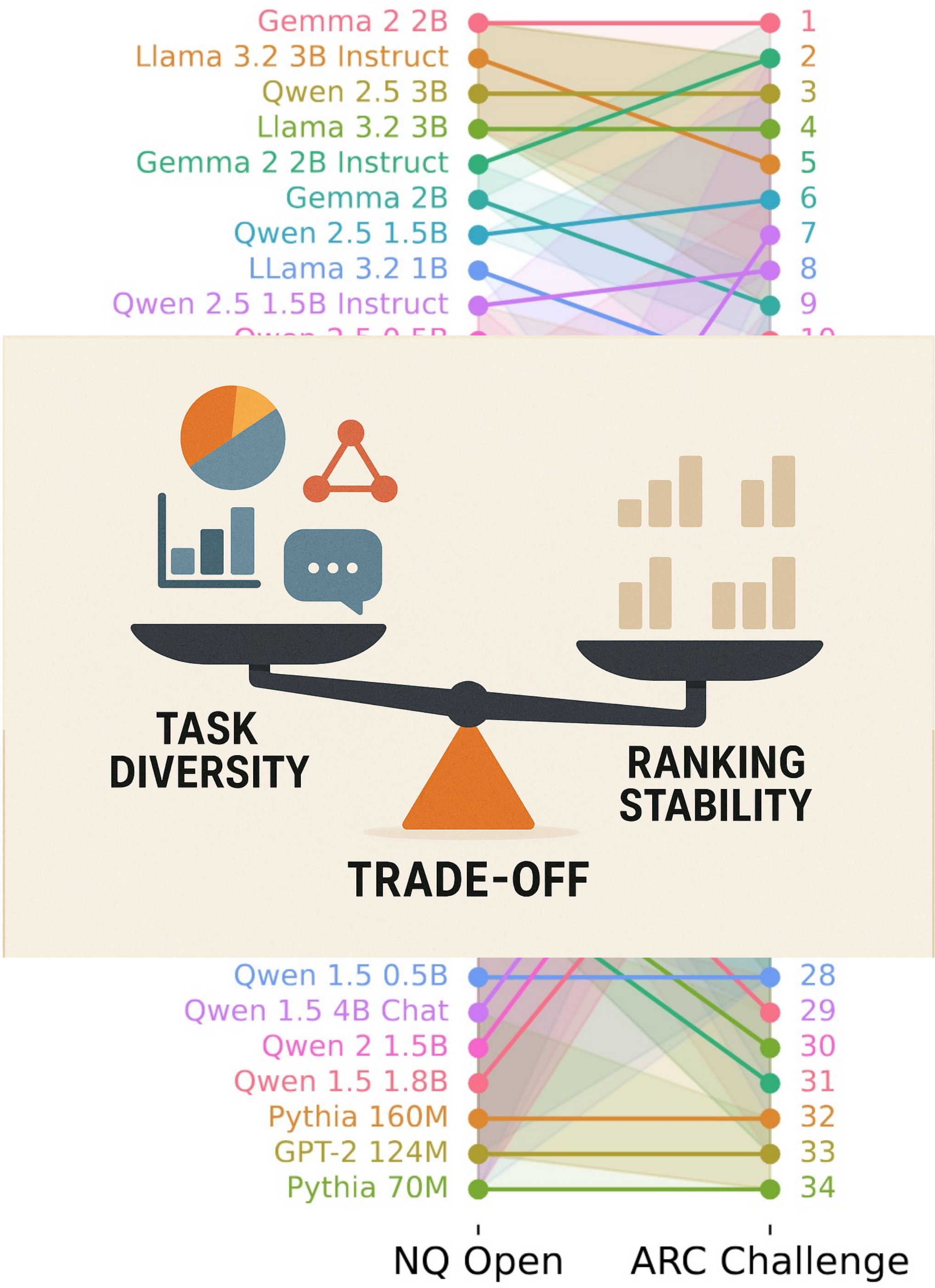


Tasks disagree with each other

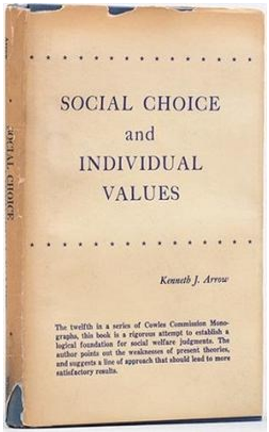
- The model rankings in different tasks often differ, even if the two tasks are similar
- *Analogy with voting system*:

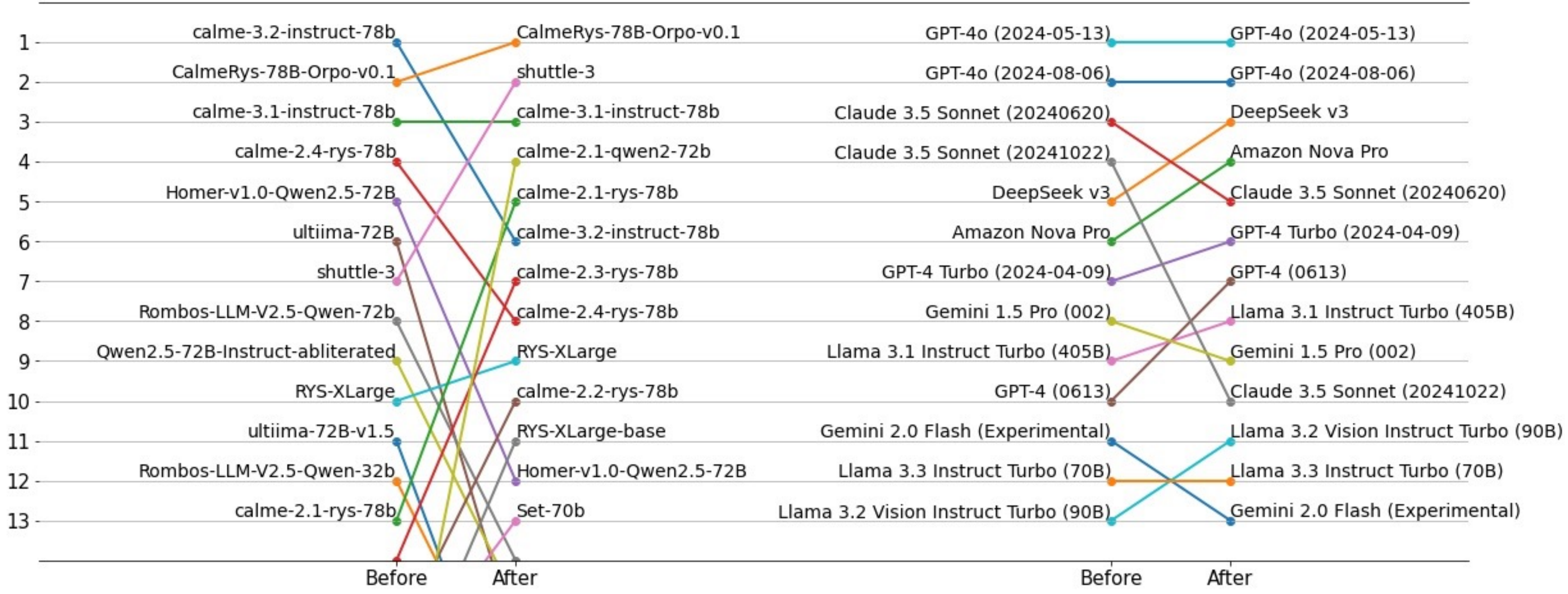
  Each task is a voter; each model is a candidate. Each voter ranks candidates

  Social choice theory: It’s hard to aggregate many rankings into one good ranking.
- Our result: Inherent trade-off between task diversity and ranking stability in multi task benchmarks



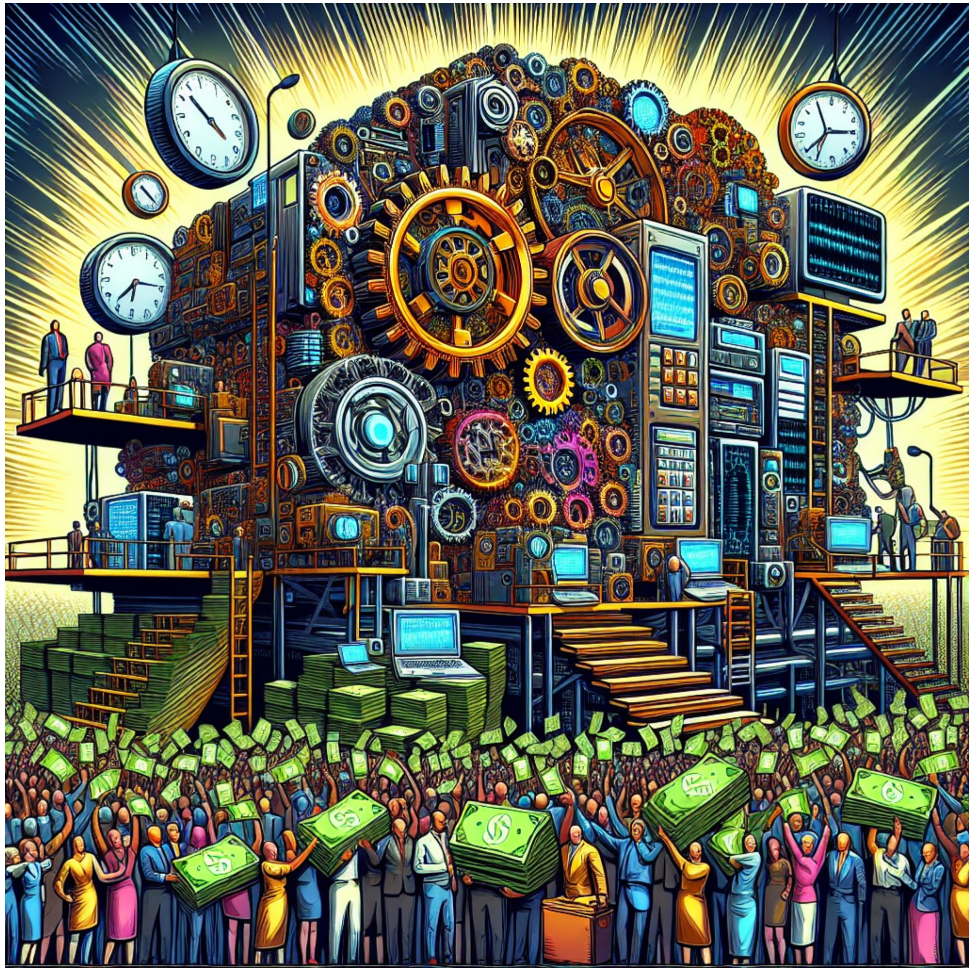
- **Sensitivity:**
    1. *Add different label noises to different tasks*
    2. *Add some irrelevant weak models*
- **Diversity:**
    - *Ranking disagreement measured by Kendall's W*





It gets worse: LLM Benchmarking is Costly

- Evaluating a single 176B parameter model, Bloom, on the HELM multi-task benchmark required 4,200 GPU hours
- People have proposed methods for benchmark performance prediction to speed up evaluation
- Our result: These methods fail at the frontier, where models are better than old models



So, it seems we're in a pinch:

1. Rankings are inconsistent
2. Computing many rankings is costly

But there's good news:

Ranking inconsistency is an artifact of how LLMs were trained

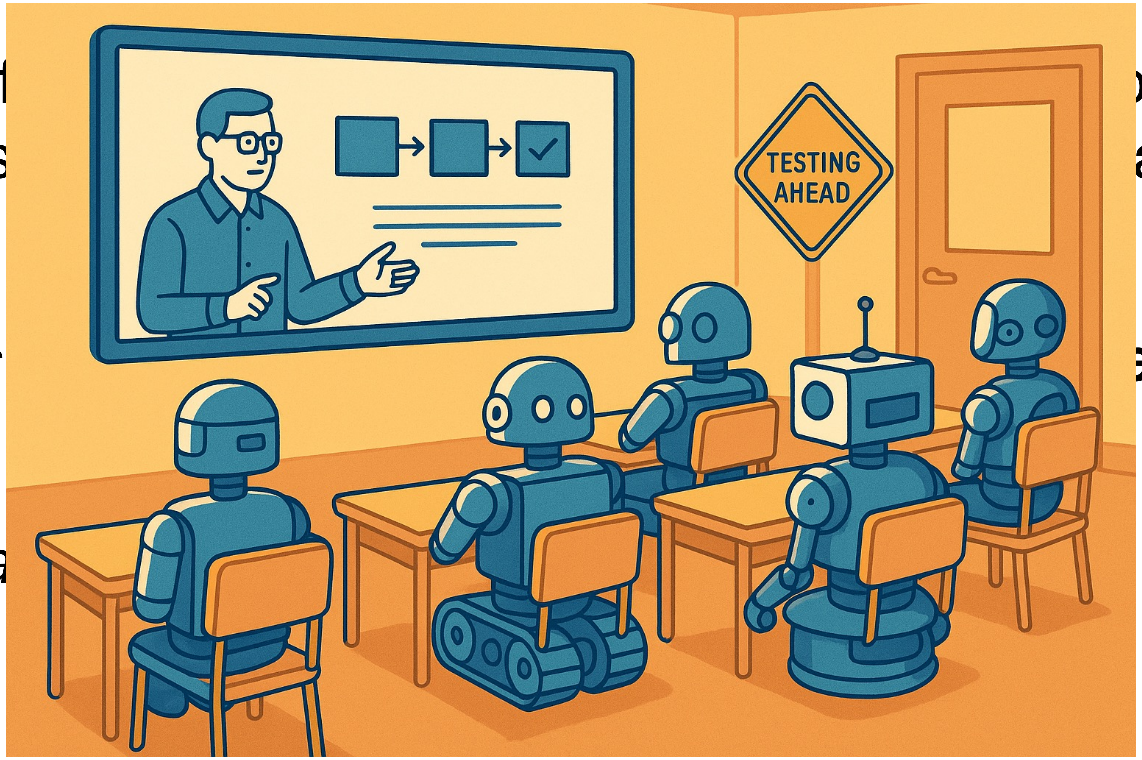
We can remove this artifact and recover highly consistent rankings

Ranking inconsistency is due to training on the test task

As released, dif f preparation
for any given tes ardt (2025)]

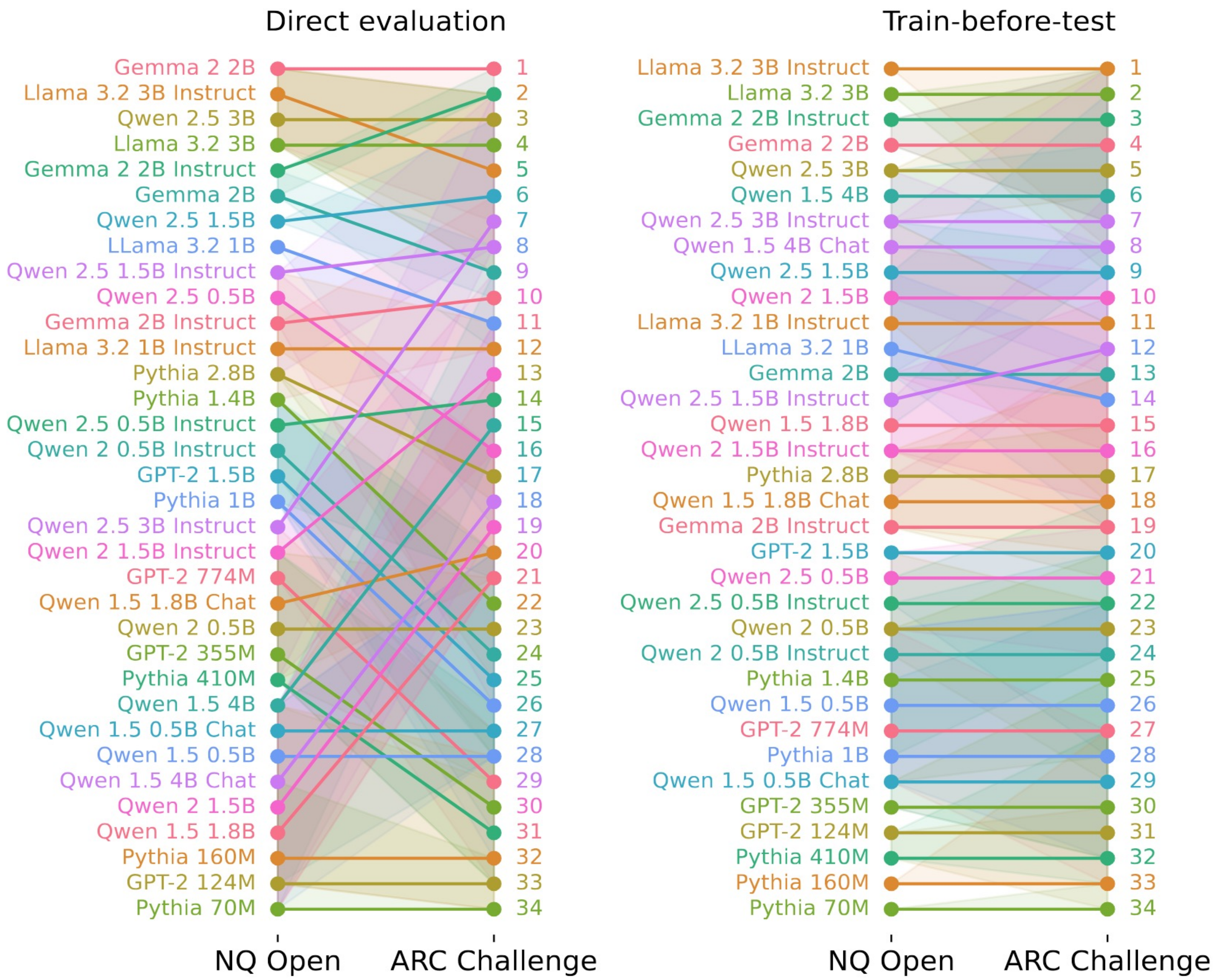
"Some models en't"

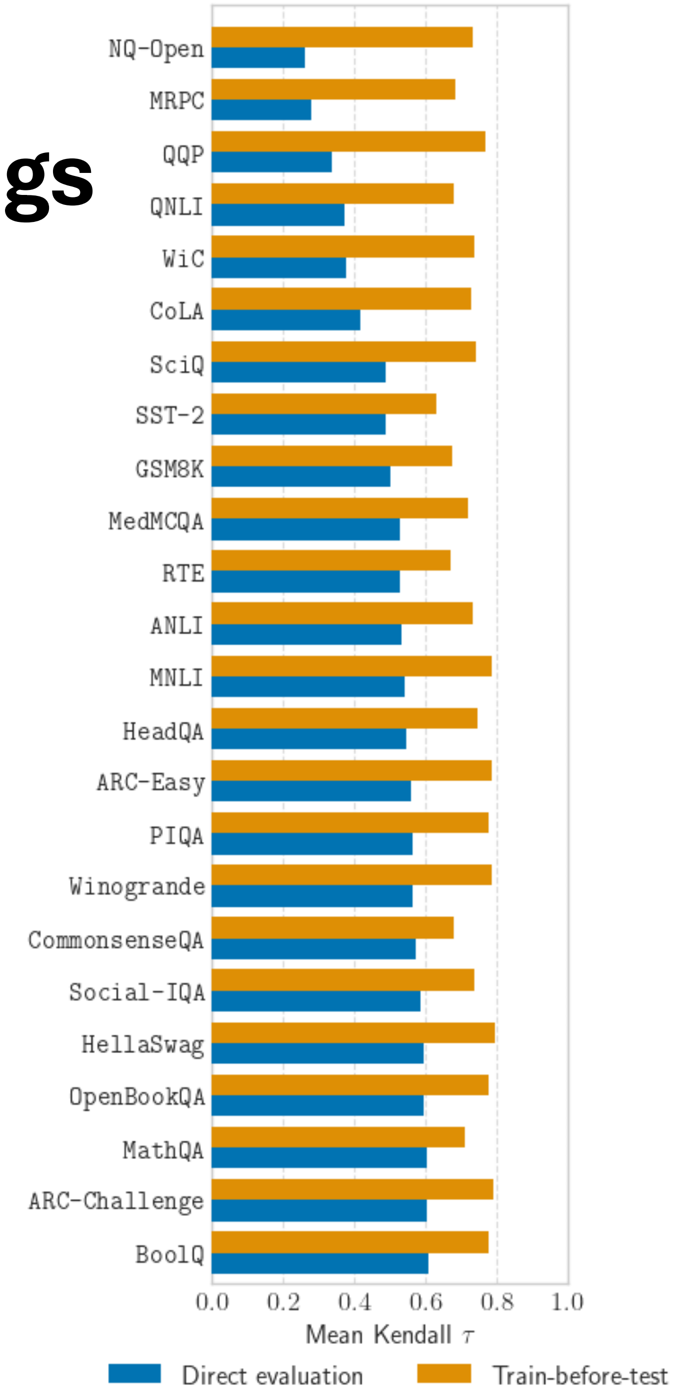
This is called *tra*



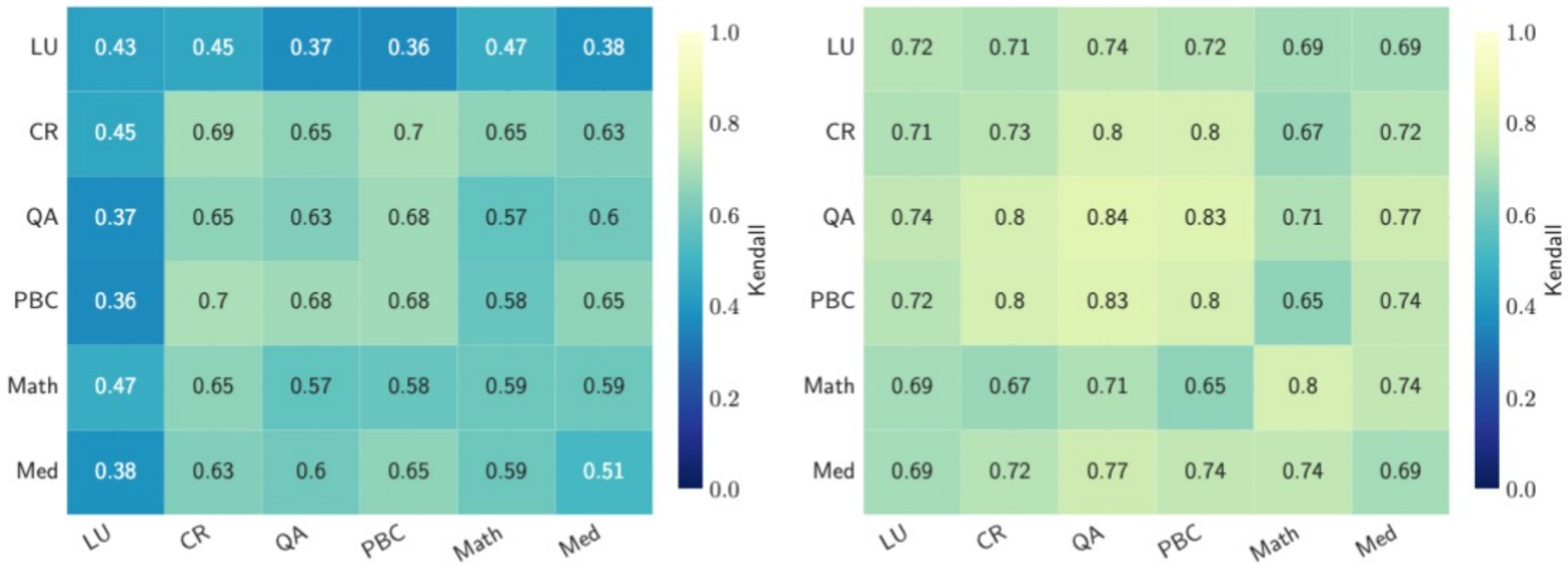
Train-before-test: Give each model the same benchmark specific fine-tuning before evaluation.

Train-before-test harmonizes model rankings





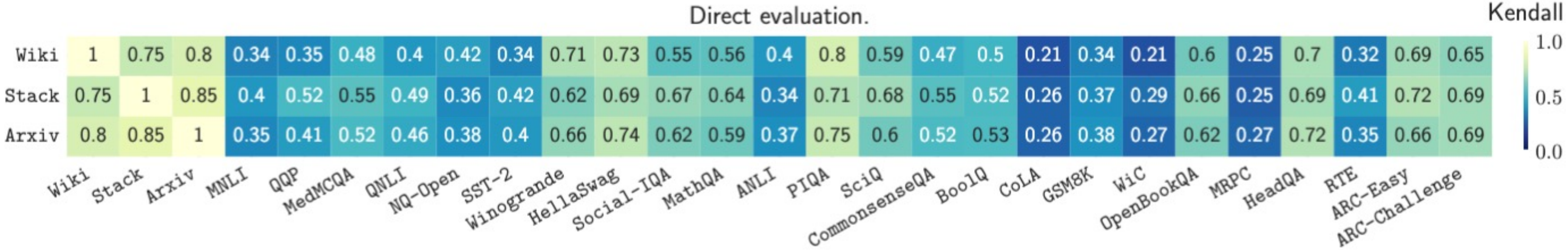
Tasks from the same category still disagree, unless …

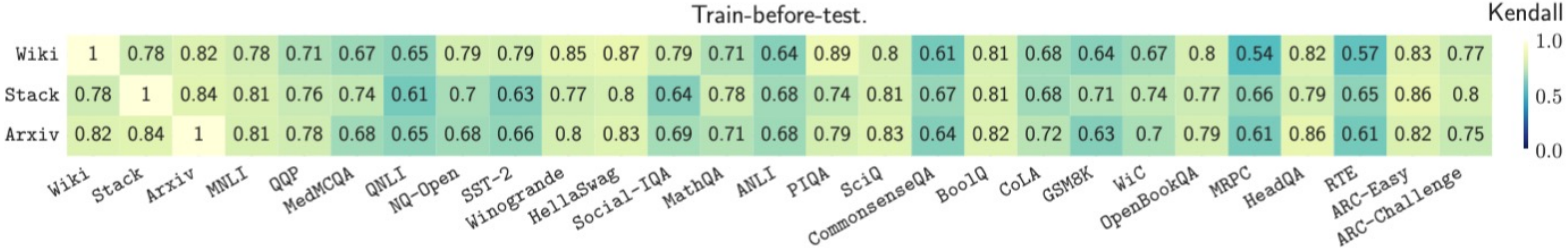


(a) Direct evaluation. (b) Train-before-test.

Figure 3: Cross-category ranking agreement for direct evaluation (left) and train-before-test (right). We consider language understanding (LU), commonsense reasoning (CR), question answering (QA), physics/biology/chemistry (PBC), math (Math), and medicine (Med) categories. Kendall's τ is averaged across all pairs of benchmarks that belong to two given categories. The diagonal represents the intra-category agreement and the others represent the inter-category agreement. train-before-test improves both intra- and inter-category ranking agreement in all instances.

Downstream agrees with perplexity under TbT

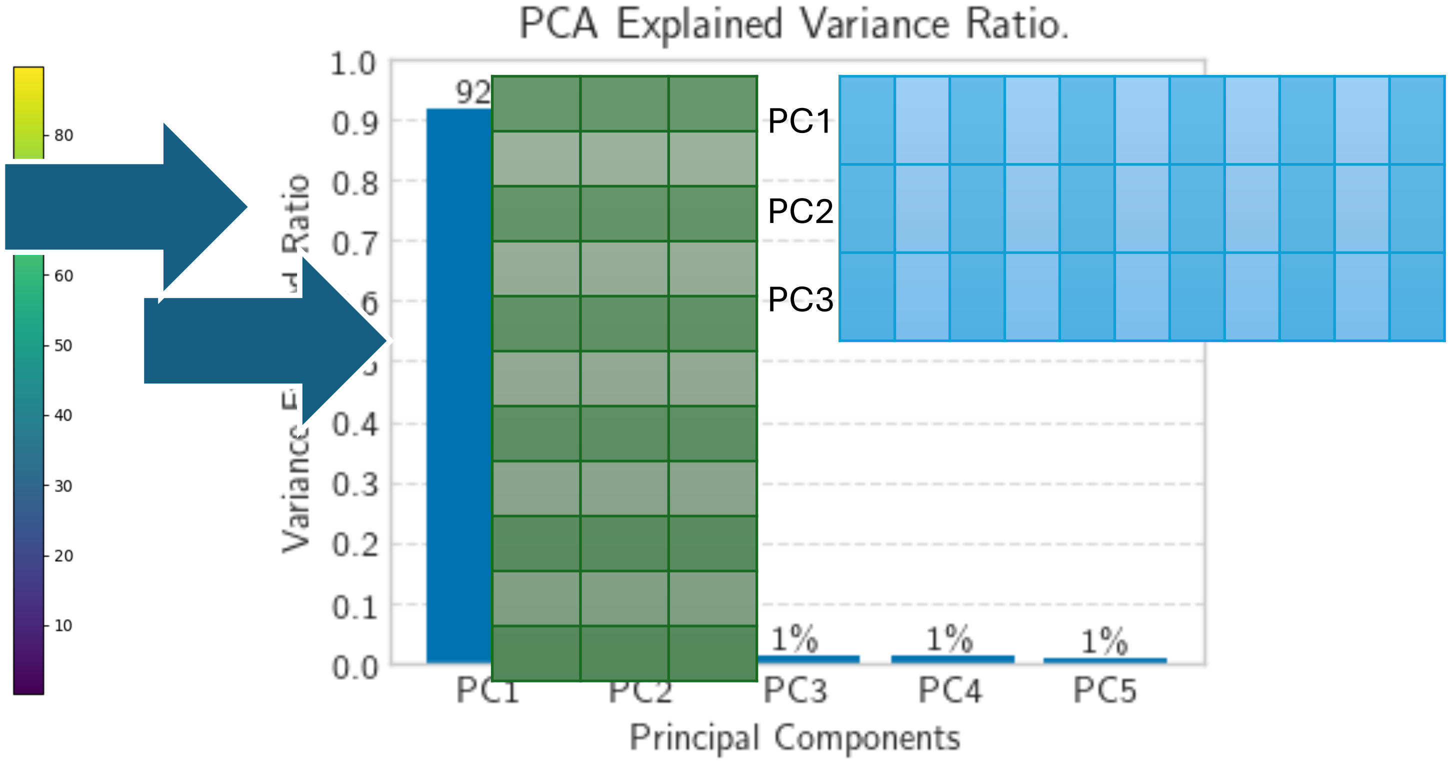




Train-before-test makes score matrix rank one

- Conduct principal component analysis (PCA) on the multi-task score matrix.

	MNLI	Winogrande	HellaSwag	GSM8K	OpenBookQA
Qwen2-7B	89.17	82.64	82.01	74.98	53.8
Qwen1.5-7B	88.9	79.08	79.52	57.24	52.2
Qwen2.5-7B	89.76	80.03	81.82	75.82	56.4
Qwen1.5-1.8B	83.88	65.75	63.58	37.07	41.4
Qwen2-1.5B	85.85	70.56	68.94	52.24	41.8
Qwen2.5-1.5B	86.48	71.19	70.16	63.15	45.4
Qwen2.5-3B	88.58	75.77	76.18	68.76	48.6
Qwen1.5-4B	87.77	74.82	74.43	52.84	45.4
Qwen2.5-0.5B	81.52	60.38	54.19	33.97	35.6
Qwen1.5-0.5B	79.6	60.14	51.35	22.82	34.2
Qwen2-0.5B	79.21	59.98	50.87	33.97	34.8



- Our result: A single factor (PC1) dominates model performances on 24 tasks

PC1 correlates with model scale

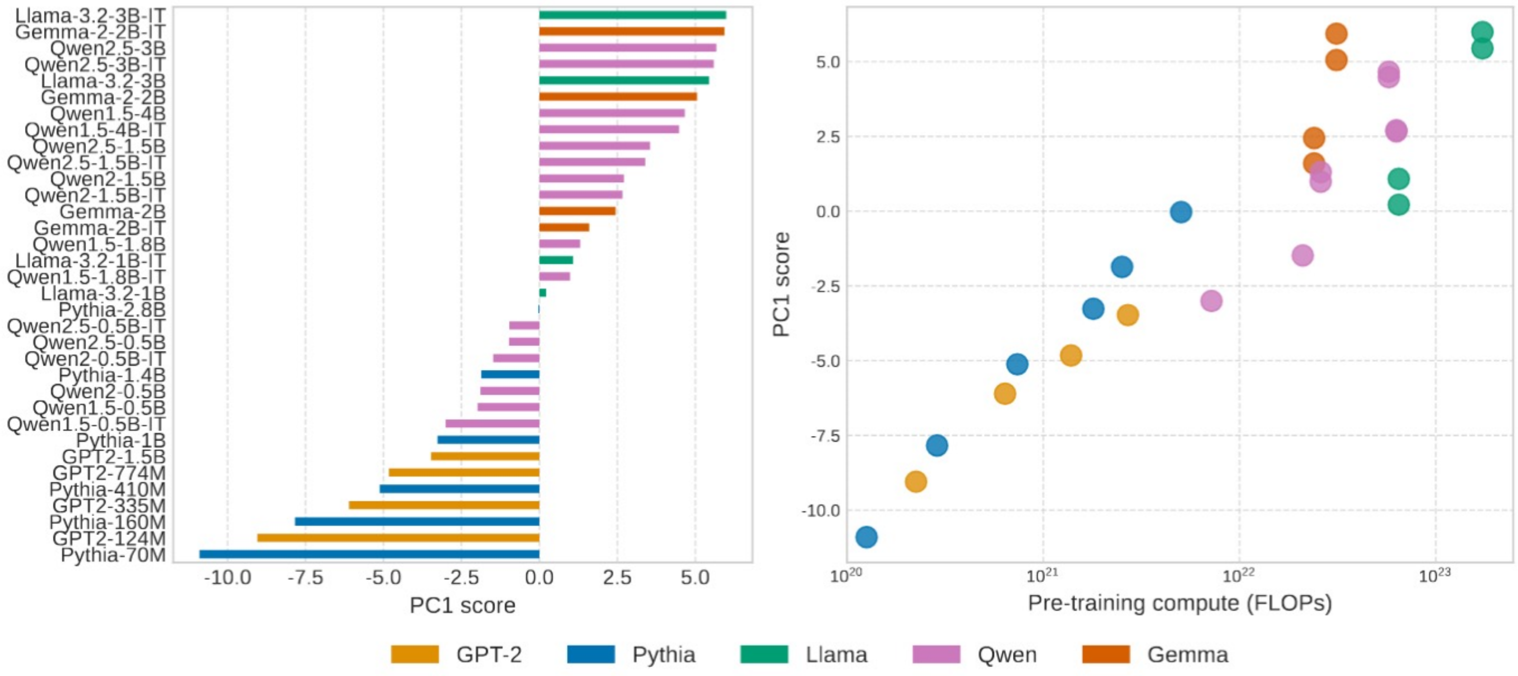
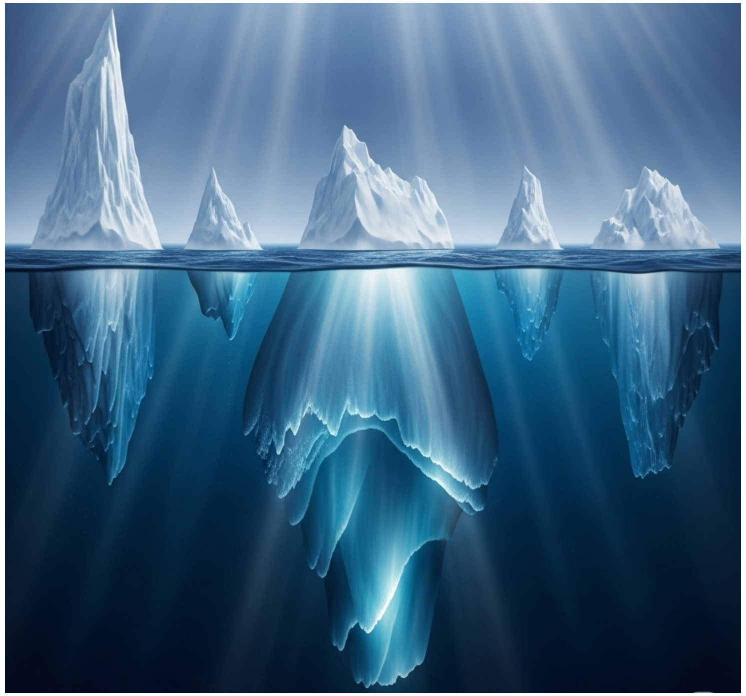


Figure 7: PC1 scores under train-before-test correlates with scale and pre-training compute.

- PC1 score stands for something **useful for all tasks**.
  - *All dimensions of PC1 is positive.*

Model potential is what really matters

- Train-before-Test measures **model potential** after development
  - Model potential rankings in any benchmark extend to others
  - Model potential correlates with perplexity of models
  - Model potential is of rank one



Take-away

Ranking is all you need
Currently benchmarking is broken for LLMs
But there’s a fix: Use train-before-test.

Thanks!