EMNLP 2024

Collaborative Performance Prediction for Large Language Models

Qiyuan Zhang[1], Fuyuan Lyu[2,3],
Xue Liu[2], Chen Ma[1]

[1]City University of Hong Kong, [2]McGill University, [3]Mila

Slides adapted from the conference presentation version made by Qiyuan Zhang

香港城市大學
City University of Hong Kong

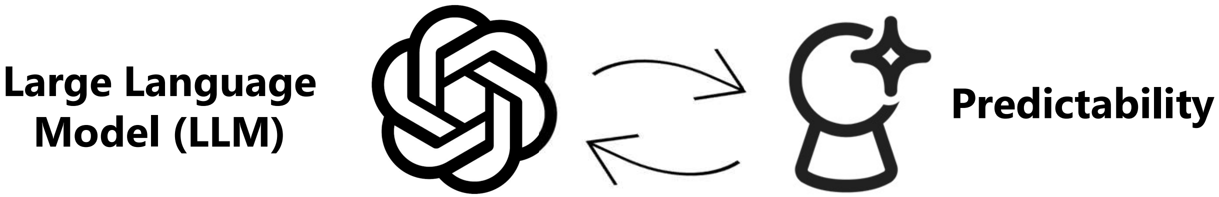
McGill UNIVERSITY

Mila

[1] GPT-4 Technical Report. arxiv.2303.08774

[2] Emergent Abilities of Large Language Models. TMLR.2022

Large Language Model

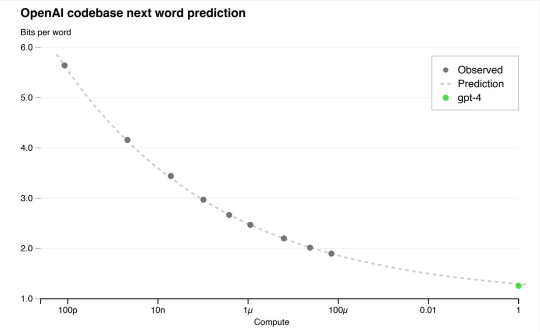


Large Language Model (LLM)

Predictability

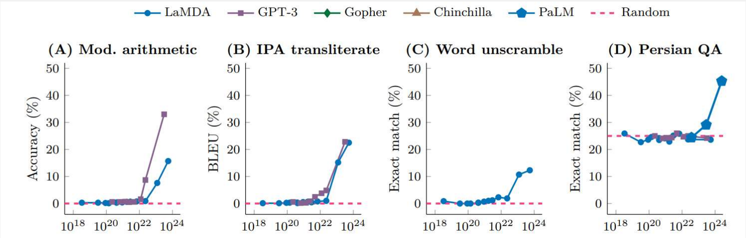
Research about Predictability

- Model Design[1]

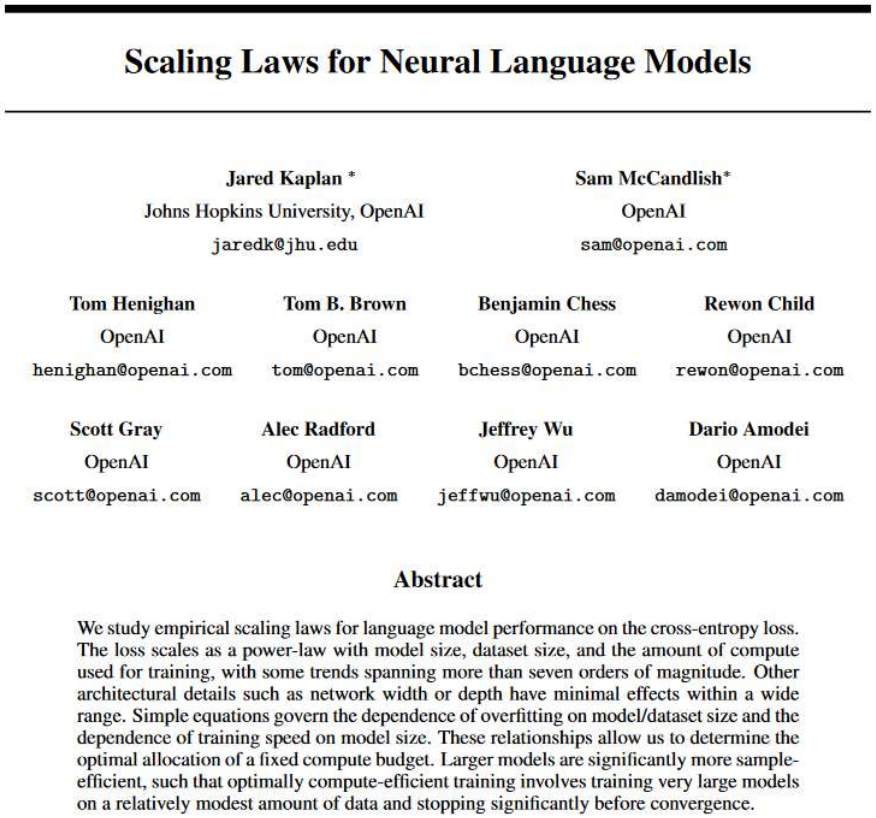


……

- Emergent Abilities[2]



Model design: “Scaling Laws for Neural Language Model”



Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

Abstract

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.

Arxiv paper in 2020

Intuition between all LLMs

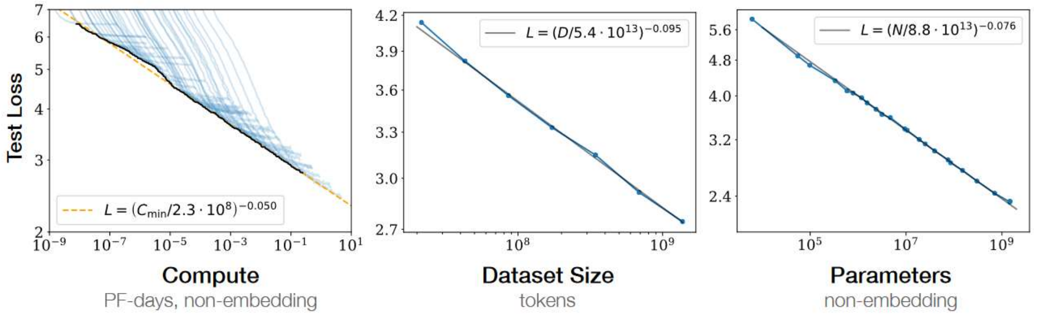


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Predictability on Downstream Tasks

"Scaling Law" is dominant method

Dominant Method

"Scaling Law"

hypothesized *power-law relationship* $\log(L_m) \approx \omega_f \log(C_m) + b_f$,

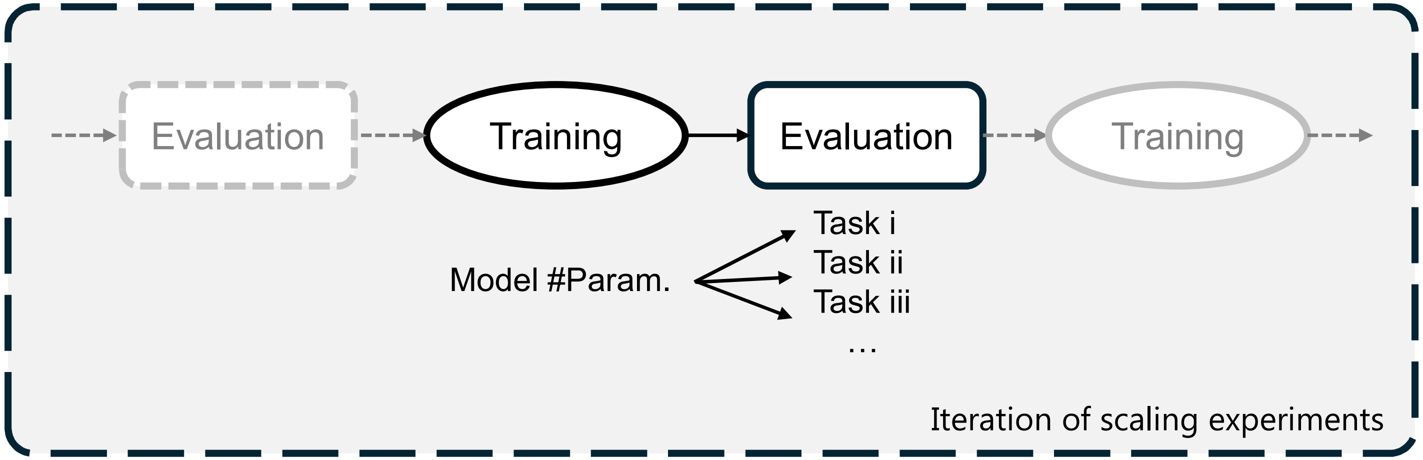
C_m : a model's computational measures, *e.g.*, training FLOPs.

L_m : their performance loss, *e.g.*, perplexity.

f : model family, *e.g.*, Llama-2 7B, 13B, and 70B

ω_f and b_f : scaling coefficients customized for each model family.

Fit this formula through repeated scaling experiments, then predict larger-scale (*C′ > C*) model.



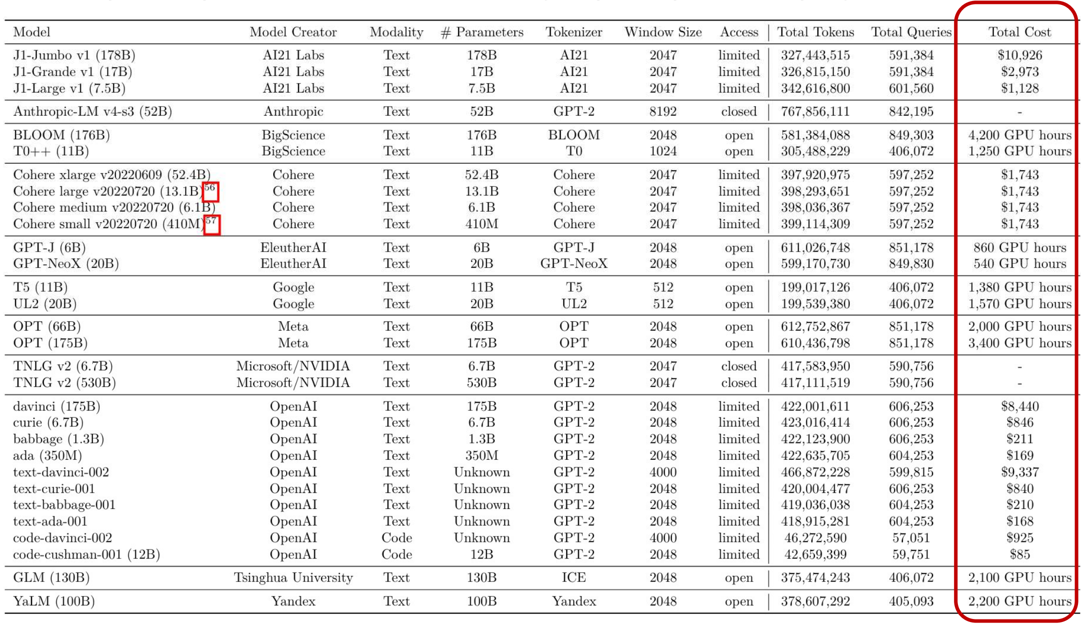
Challenges

"Scaling Law'' is not enough

1. High Cost

Training Cost[3]: repeated scaling training models (1B, 8B, 70B) in a family.

Inference Cost[4]: Testing various models in various benchmarks, especially for scaled models (>70B) and Chain-of-Thought(CoT) tasks (e.g., Math Reasoning).



Model	Model Creator	Modality	# Parameters	Tokenizer	Window Size	Access	Total Tokens	Total Queries	Total Cost
J1-Jumbo v1 (178B)	AI21 Labs	Text	178B	AI21	2047	limited	327,443,515	591,384	$10,926
J1-Grande v1 (17B)	AI21 Labs	Text	17B	AI21	2047	limited	326,815,150	591,384	$2,973
J1-Large v1 (7.5B)	AI21 Labs	Text	7.5B	AI21	2047	limited	342,616,800	601,560	$1,128
Anthropic-LM v4-s3 (52B)	Anthropic	Text	52B	GPT-2	8192	closed	767,856,111	842,195	-
BLOOM (176B)	BigScience	Text	176B	BLOOM	2048	open	581,384,088	849,303	4,200 GPU hours
T0++ (11B)	BigScience	Text	11B	T0	1024	open	305,488,229	406,072	1,250 GPU hours
Cohere xlarge v20220609 (52.4B)	Cohere	Text	52.4B	Cohere	2047	limited	397,920,975	597,252	$1,743
Cohere large v20220720 (13.1B)[56]	Cohere	Text	13.1B	Cohere	2047	limited	398,293,651	597,252	$1,743
Cohere medium v20220720 (6.1B)	Cohere	Text	6.1B	Cohere	2047	limited	398,036,367	597,252	$1,743
Cohere small v20220720 (410M)[57]	Cohere	Text	410M	Cohere	2047	limited	399,114,309	597,252	$1,743
GPT-J (6B)	EleutherAI	Text	6B	GPT-J	2048	open	611,026,748	851,178	860 GPU hours
GPT-NeoX (20B)	EleutherAI	Text	20B	GPT-NeoX	2048	open	599,170,730	849,830	540 GPU hours
T5 (11B)	Google	Text	11B	T5	512	open	199,017,126	406,072	1,380 GPU hours
UL2 (20B)	Google	Text	20B	UL2	512	open	199,539,380	406,072	1,570 GPU hours
OPT (66B)	Meta	Text	66B	OPT	2048	open	612,752,867	851,178	2,000 GPU hours
OPT (175B)	Meta	Text	175B	OPT	2048	open	610,436,798	851,178	3,400 GPU hours
TNLG v2 (6.7B)	Microsoft/NVIDIA	Text	6.7B	GPT-2	2047	closed	417,583,950	590,756	-
TNLG v2 (530B)	Microsoft/NVIDIA	Text	530B	GPT-2	2047	closed	417,111,519	590,756	-
davinci (175B)	OpenAI	Text	175B	GPT-2	2048	limited	422,001,611	606,253	$8,440
curie (6.7B)	OpenAI	Text	6.7B	GPT-2	2048	limited	423,016,414	606,253	$846
babbage (1.3B)	OpenAI	Text	1.3B	GPT-2	2048	limited	422,123,900	606,253	$211
ada (350M)	OpenAI	Text	350M	GPT-2	2048	limited	422,635,705	604,253	$169
text-davinci-002	OpenAI	Text	Unknown	GPT-2	4000	limited	466,872,228	599,815	$9,337
text-curie-001	OpenAI	Text	Unknown	GPT-2	2048	limited	420,004,477	606,253	$840
text-babbage-001	OpenAI	Text	Unknown	GPT-2	2048	limited	419,036,038	604,253	$210
text-ada-001	OpenAI	Text	Unknown	GPT-2	2048	limited	418,915,281	604,253	$168
code-davinci-002	OpenAI	Code	Unknown	GPT-2	4000	limited	46,272,590	57,051	$925
code-cushman-001 (12B)	OpenAI	Code	12B	GPT-2	2048	limited	42,659,399	59,751	$85
GLM (130B)	Tsinghua University	Text	130B	ICE	2048	open	375,474,243	406,072	2,100 GPU hours
YaLM (100B)	Yandex	Text	100B	Yandex	2048	open	378,607,292	405,093	2,200 GPU hours

$10K+ and 4K+ GPU hours

Figure 1. Inference Cost of each model in HELM Benchmark.

[3] Predicting Emergent Abilities with Infinite Resolution Evaluation. ICLR 2023

[4] Holistic Evaluation of Language Models. TMLR 2023

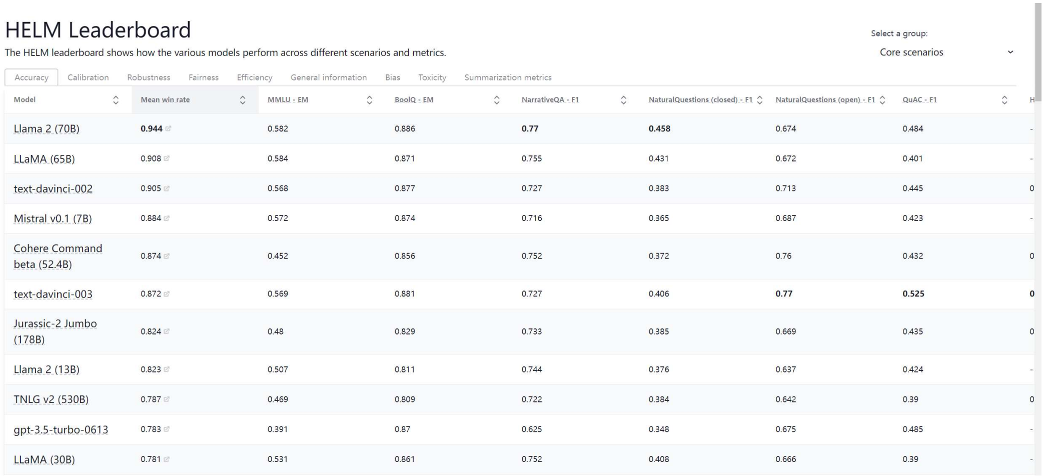
[5] Textbooks Are All You Need. Arxiv. 2306.11644

[6] CompassBench. https://opencompass.org.cn/doc

2. Missing other factors

Scaling law only consider *computational measures* factor but ignore many important factors, e.g., *Data Quality* [5], *Model Hyperparameters*, ….

3. Ignore relationship among models and tasks.

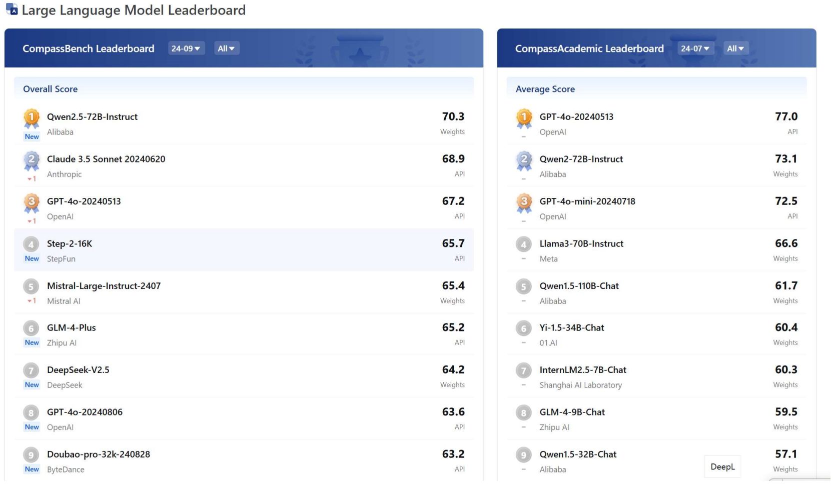


HELM Leaderboard

The HELM leaderboard shows how the various models perform across different scenarios and metrics.

Select a group: Core scenarios

Model	Mean win rate	MMLU - EM	BoolQ - EM	NarrativeQA - F1	NaturalQuestions (closed) - F1	NaturalQuestions (open) - F1	QuAC - F1	H
Llama 2 (70B)	**0.944**	0.582	0.886	**0.77**	**0.458**	0.674	0.484	-
LLaMA (65B)	0.908	0.584	0.871	0.755	0.431	0.672	0.401	-
text-davinci-002	0.905	0.568	0.877	0.727	0.383	0.713	0.445	0
Mistral v0.1 (7B)	0.884	0.572	0.874	0.716	0.365	0.687	0.423	-
Cohere Command beta (52.4B)	0.874	0.452	0.856	0.752	0.372	0.76	0.432	0
text-davinci-003	0.872	0.569	0.881	0.727	0.406	**0.77**	**0.525**	**0**
Jurassic-2 Jumbo (178B)	0.824	0.48	0.829	0.733	0.385	0.669	0.435	0
Llama 2 (13B)	0.823	0.507	0.811	0.744	0.376	0.637	0.424	-
TNLG v2 (530B)	0.787	0.469	0.809	0.722	0.384	0.642	0.39	0
gpt-3.5-turbo-0613	0.783	0.391	0.87	0.625	0.348	0.675	0.485	-
LLaMA (30B)	0.781	0.531	0.861	0.752	0.408	0.666	0.39	-



Large Language Model Leaderboard

CompassBench Leaderboard 24-09 All

Overall Score

1	New	Qwen2.5-72B-Instruct	Alibaba	70.3	Weights
2	▾1	Claude 3.5 Sonnet 20240620	Anthropic	68.9	API
3	▾1	GPT-4o-20240513	OpenAI	67.2	API
4	New	Step-2-16K	StepFun	65.7	API
5	▾1	Mistral-Large-Instruct-2407	Mistral AI	65.4	Weights
6	New	GLM-4-Plus	Zhipu AI	65.2	API
7	New	DeepSeek-V2.5	DeepSeek	64.2	Weights
8	New	GPT-4o-20240806	OpenAI	63.6	API
9	New	Doubao-pro-32k-240828	ByteDance	63.2	API

CompassAcademic Leaderboard 24-07 All

Average Score

1	-	GPT-4o-20240513	OpenAI	77.0	API
2	-	Qwen2-72B-Instruct	Alibaba	73.1	Weights
3	-	GPT-4o-mini-20240718	OpenAI	72.5	API
4	-	Llama3-70B-Instruct	Meta	66.6	Weights
5	-	Qwen1.5-110B-Chat	Alibaba	61.7	Weights
6	-	Yi-1.5-34B-Chat	01.AI	60.4	Weights
7	-	InternLM2.5-7B-Chat	Shanghai AI Laboratory	60.3	Weights
8	-	GLM-4-9B-Chat	Zhipu AI	59.5	Weights
9	-	Qwen1.5-32B-Chat	Alibaba	57.1	Weights

--[6]

Pros & Cons of Scaling Law

A Summary of Scaling Law

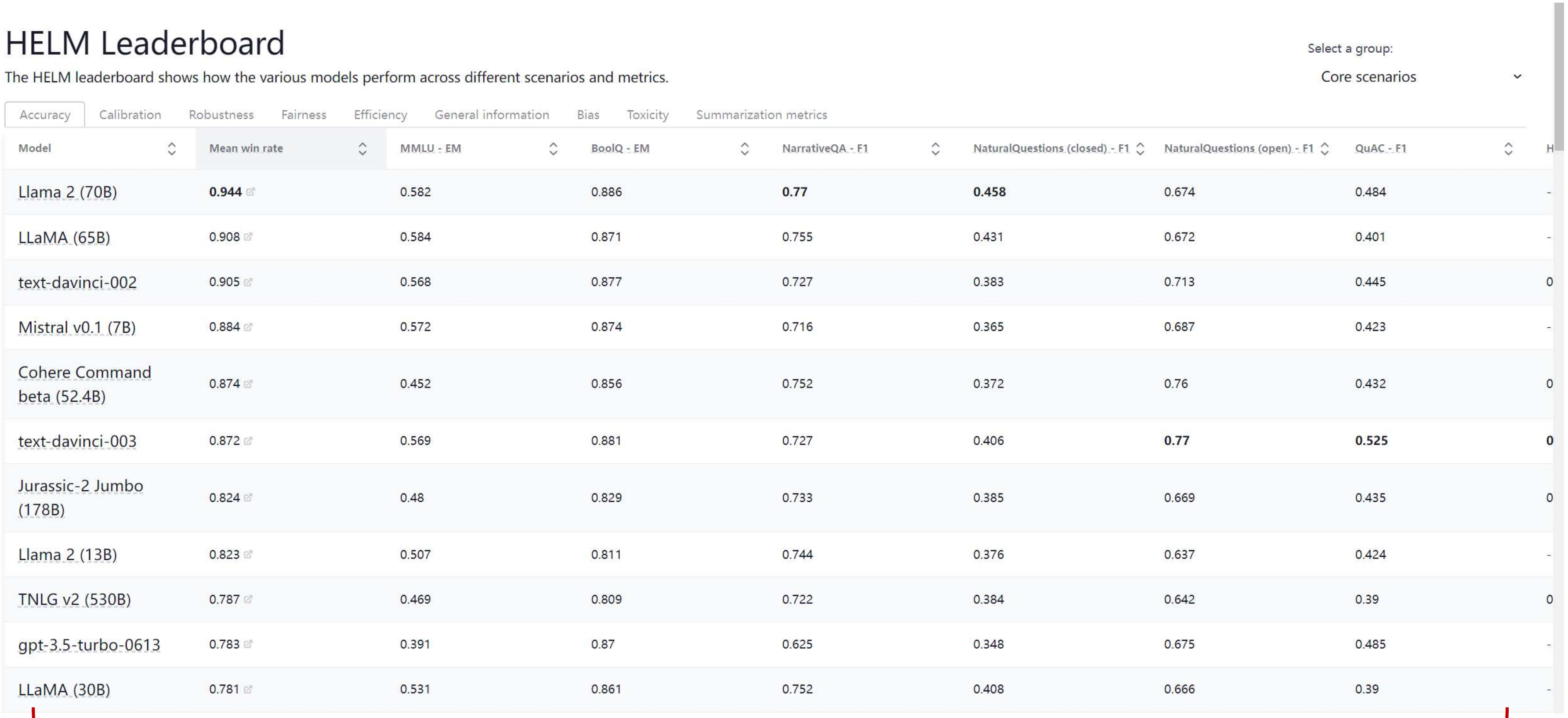
$$\log(L_m) \approx \omega_f \log(C_m) + \ b_f \ ,$$

1. There exists predictability in LLMs.
2. Predictability is limited to one single model family.
3. Predictability is limited to one metric.
4. The fitting of the scaling law is cost.
5. Inference needs inputting transparent design factors.
6. Neglecting other possible factors, e.g., data quality.

If predict the performance of LLMs on downstream tasks,
what other methods can we use **beyond scaling laws**?

Beyond Scaling Law

If predict the performance of LLMs on downstream tasks,
what other methods can we use beyond scaling laws?



HELM Leaderboard

The HELM leaderboard shows how the various models perform across different scenarios and metrics.

Model	Mean win rate	MMLU - EM	BoolQ - EM	NarrativeQA - F1	NaturalQuestions (closed) - F1	NaturalQuestions (open) - F1	QuAC - F1	H
Llama 2 (70B)	**0.944**	0.582	0.886	**0.77**	**0.458**	0.674	0.484	-
LLaMA (65B)	0.908	0.584	0.871	0.755	0.431	0.672	0.401	-
text-davinci-002	0.905	0.568	0.877	0.727	0.383	0.713	0.445	0
Mistral v0.1 (7B)	0.884	0.572	0.874	0.716	0.365	0.687	0.423	-
Cohere Command beta (52.4B)	0.874	0.452	0.856	0.752	0.372	0.76	0.432	0
text-davinci-003	0.872	0.569	0.881	0.727	0.406	**0.77**	**0.525**	**0**
Jurassic-2 Jumbo (178B)	0.824	0.48	0.829	0.733	0.385	0.669	0.435	0
Llama 2 (13B)	0.823	0.507	0.811	0.744	0.376	0.637	0.424	-
TNLG v2 (530B)	0.787	0.469	0.809	0.722	0.384	0.642	0.39	0
gpt-3.5-turbo-0613	0.783	0.391	0.87	0.625	0.348	0.675	0.485	-
LLaMA (30B)	0.781	0.531	0.861	0.752	0.408	0.666	0.39	-

Matrix Factorization?

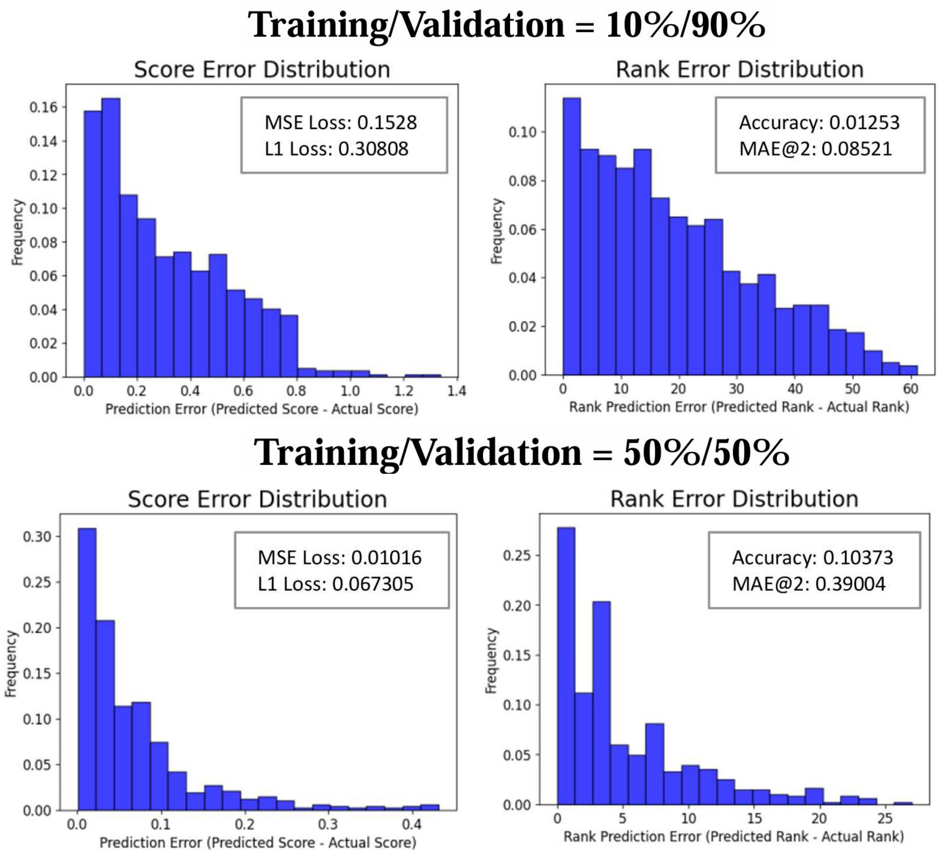
Pilot Demonstration

Matrix Factorization on HELM Leaderboard (Open-source)

- HELM Core Leaderboard
  -- 68 models and 16 tasks, a score matrix with a density of 82.5%

- Matrix Factorization (MF)
  -- # Factor = 10

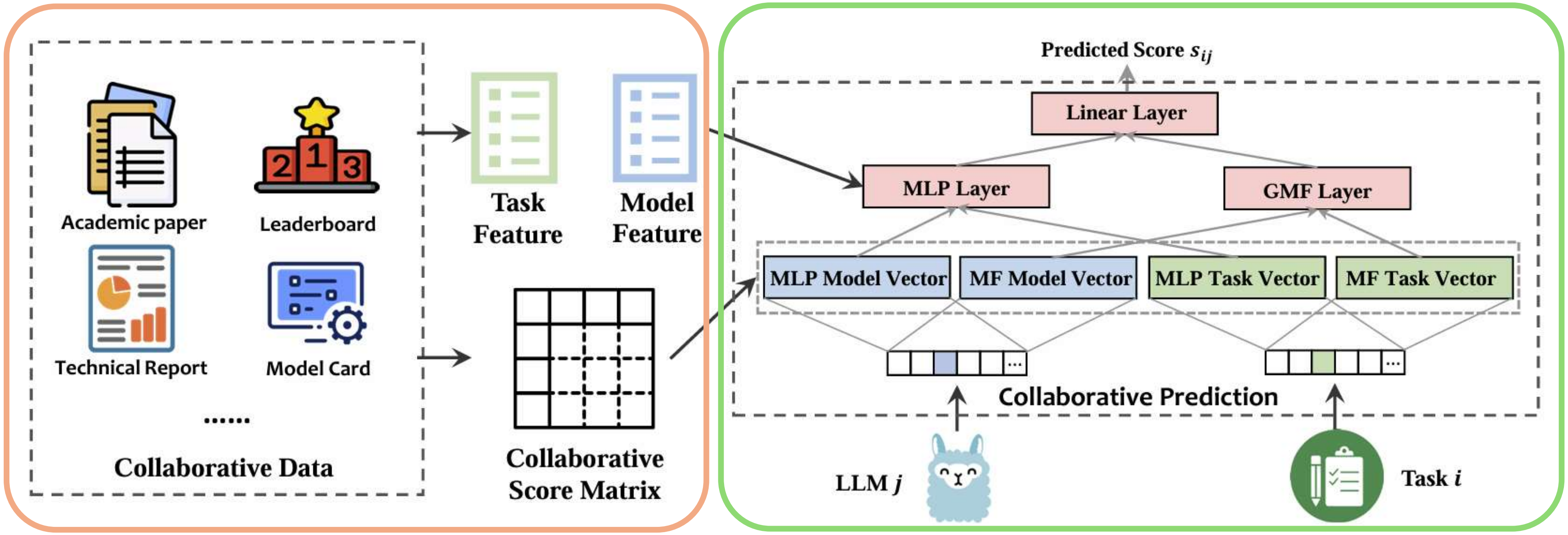
Conclusion: MF can accurately predict most of the missing scores within a low error range.


Training/Validation = 10%/90%

Training/Validation = 50%/50%

Figure 2. Error Distribution of Predictions based on the open-source Leaderboard Using Matrix Factorization.

Collaborative Performance Prediction



(i) Collaborative performance data

(ii) Collaborative prediction methods

Comparison

Cons of Scaling Law

$$\log(L_m) \approx \omega_f \log(C_m) + \ b_f \ ,$$

1. Predictability is limited to one single model family.
2. Predictability is limited to one metric in one task.
3. The fitting of the scaling law is cost.
4. Inference needs inputting transparent design factors.
5. Neglecting other possible factors, e.g., data quality.

Collaborative Performance Prediction (CPP)

1. Predictability supports cross model-families.
2. Predictability supports cross tasks.
3. Low Training Cost.
4. Predictability supports proprietary model.
5. Predictability supports more factors beyond scaling law.
6. Factor-level Interpretability.

Collaborative Data

We support any score matrixes, including open-source leaderboards and custom leaderboards.

- **Open-source Leaderboard**

  HELM, OpenLLM[7], Compass

  Sparsity < 15%

- **Custom Leaderboard**

  3 Leaderboard

  55 Paper/Technical Report

  31 Model Card

  Collect the collaborative performances

  #Models = 72
  #Tasks = 22
  #Model Features = 16
  #Task Features = 4
  Sparsity = 44%

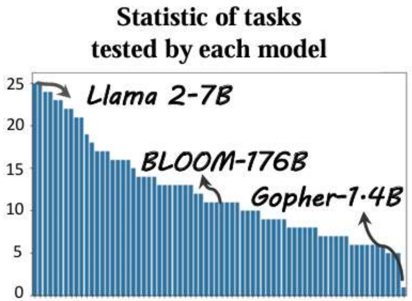
[7] OpenLLM, https://github.com/bentoml/OpenLLM

Analysis on Custom Leaderboard

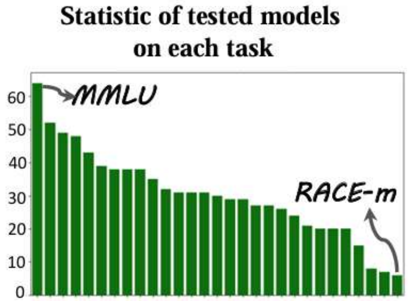
- **Uneven distribution of testing resources.**

  MMLU and HellaSwag ↔ RACE-m

  Llama 2-7B ↔ Gopher-1.4B



Statistic of tasks tested by each model



Statistic of tested models on each task

- **Widespread variations in the scores.**

  identical models yield varying scores on the same tasks across different studies.

- **Missing description/model card.** [8]

  We encourage everyone should open-source the design factors as many as possible.

[8] open-source and science in the era of foundation models. EMNLP 2024 Keynotes, Percy Liang

Collaborative Methods

Matrix Factorization & Neural Collaborative Filtering

Let M = {M_1,M_2,...,Mn} be a set of n LLMs, and T = {T_1,T_2,...,T_m} be a suite of m tasks. Define the Score Matrix S, which is an $n \times m$ matrix where each element s_{ij} represents the performance score of model M_i on task T_j. s_{ij} is defined as

$$s_{ij} = \begin{cases} \textit{score} & \textit{if tested,} \\ \textit{unknown} & \textit{otherwise}. \end{cases}$$

Neural collaborative filtering uses a multi-layer perceptron to learn the model-task interaction function to predict the score $\widehat{s}_{ij}$ for a model i on a task j,

collaborative models and tasks

Optional descriptive factors

$$\begin{aligned} \widehat{s}_{ij} &= f(i,j|\mathcal{M},\mathcal{T},[\mathcal{V}_i,\mathcal{V}_j],\theta) \\ &= \text{MLP}(p_i,q_j,[e_{vi},e_{vj}]), \end{aligned}$$

the latent vectors for model i and task j

embeddings of descriptive factors

Optionally, we can predict a score when only inputting the descriptive factors,

$$\begin{aligned} \widehat{s}_{ij} &= f(i,j|\mathcal{V}_i,\mathcal{V}_j,\theta) \\ &= \text{MLP}(e_{vi},e_{vj}), \end{aligned}$$

Loss function is

$$L(\theta) = \frac{1}{N} \sum_{(i,j)\in\mathcal{D}} (\widehat{s}_{ij} - s_{ij})^2,$$

Experiment Setting

Evaluation Metric.

Score-Based: MSE & L1 Loss (Predicted Score and Gold Normalized Score)

Rank-Based: Accuracy and MAE@2 (Rank of Predicted Scores and Gold Scores.)

$$\text{Accuracy} = \left(\frac{\sum_{i=1}^{N} \mathbf{1}(r_i = \widehat{r}_i)}{N} \right) \times 100\%, \qquad \text{MAE@2} = \left(\frac{\sum_{i=1}^{N} \mathbf{1}(|r_i - \widehat{r}_i| \leq 2)}{N} \right) \times 100\%,$$

Variation of Models.

Matrix Factorization

Neural Collaborative Filtering

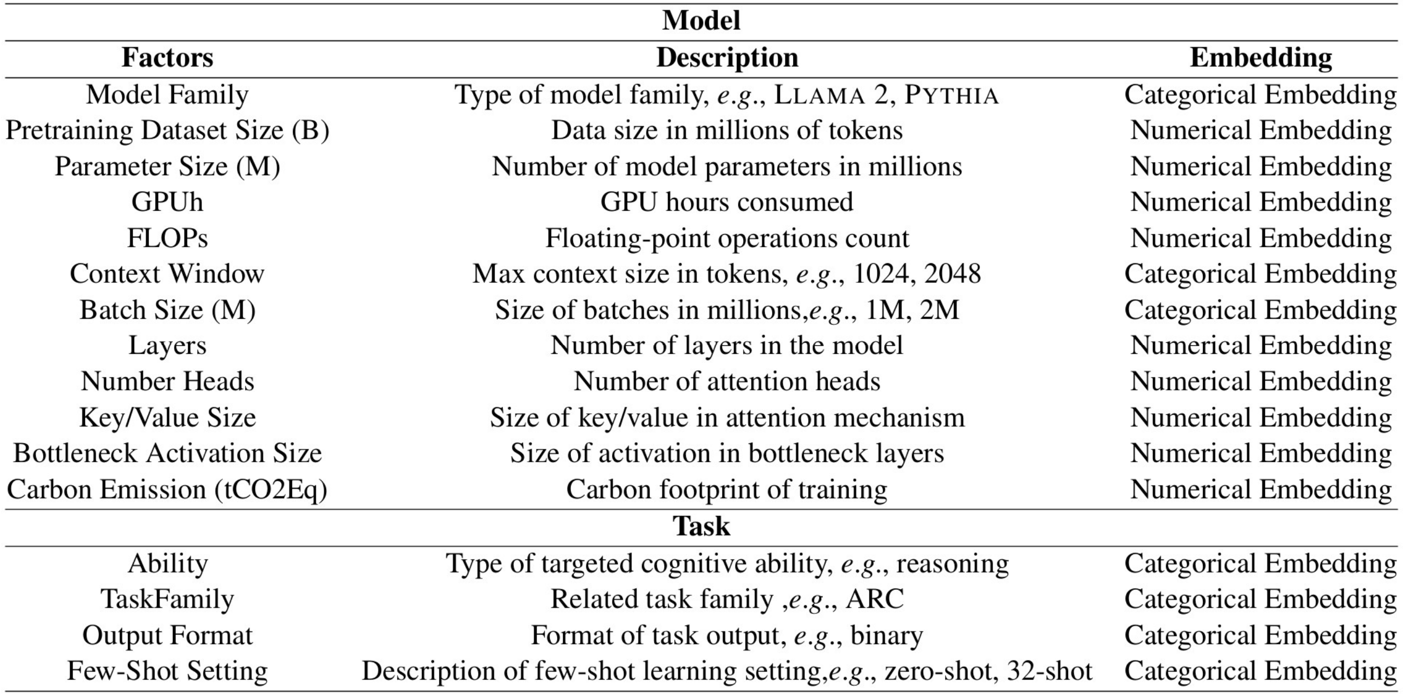
Neural Collaborative Filtering (Factor-enhanced)

Neural Collaborative Filtering (only Factor)

Model Configuration

latent factors = 10, learning rate = 0.01, iteration = 250,000

Descriptive Factors.



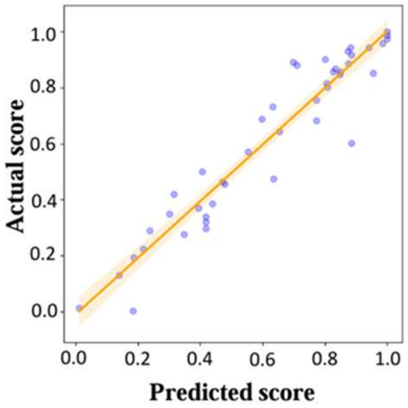
Model		
Factors	**Description**	**Embedding**
Model Family	Type of model family, *e.g.*, Llama 2, Pythia	Categorical Embedding
Pretraining Dataset Size (B)	Data size in millions of tokens	Numerical Embedding
Parameter Size (M)	Number of model parameters in millions	Numerical Embedding
GPUh	GPU hours consumed	Numerical Embedding
FLOPs	Floating-point operations count	Numerical Embedding
Context Window	Max context size in tokens, *e.g.*, 1024, 2048	Categorical Embedding
Batch Size (M)	Size of batches in millions,*e.g.*, 1M, 2M	Categorical Embedding
Layers	Number of layers in the model	Numerical Embedding
Number Heads	Number of attention heads	Numerical Embedding
Key/Value Size	Size of key/value in attention mechanism	Numerical Embedding
Bottleneck Activation Size	Size of activation in bottleneck layers	Numerical Embedding
Carbon Emission (tCO2Eq)	Carbon footprint of training	Numerical Embedding
Task		
Ability	Type of targeted cognitive ability, *e.g.*, reasoning	Categorical Embedding
TaskFamily	Related task family ,*e.g.*, ARC	Categorical Embedding
Output Format	Format of task output, *e.g.*, binary	Categorical Embedding
Few-Shot Setting	Description of few-shot learning setting,*e.g.*, zero-shot, 32-shot	Categorical Embedding

Partition.

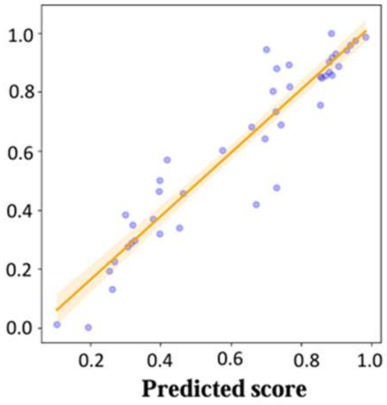
Validation Set = 5%, because the sparsity of the original matrix is 44%.

Main Result

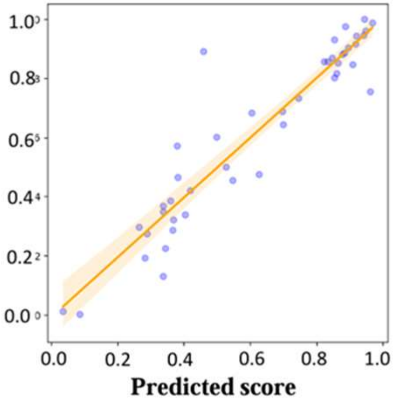
- Collaborative Filtering Mechanisms is Feasible.



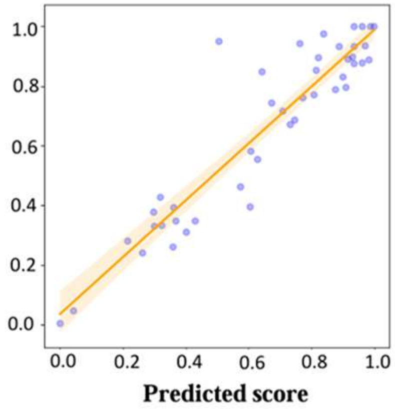
Matrix Factorization



Neural Collaborative Filtering

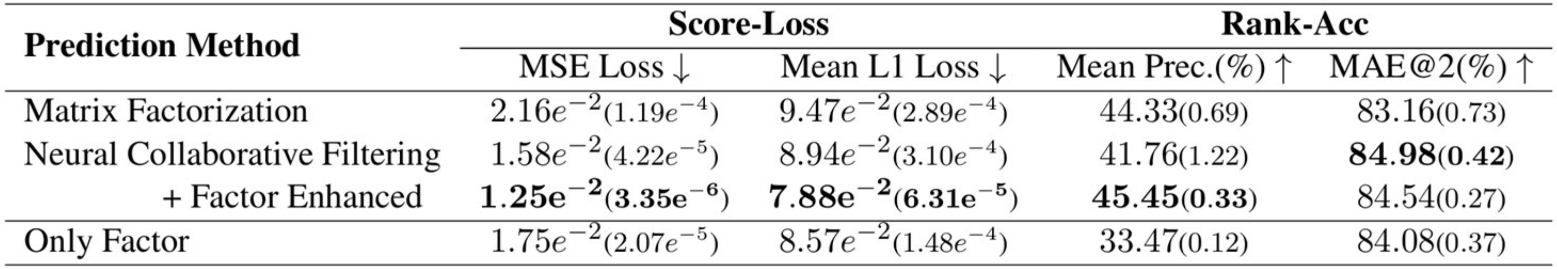


Neural Collaborative Filtering (Factor-enhanced)



Neural Collaborative Filtering (only Factor)

Predicted Score ≈ Gold Score



Prediction Method	Score-Loss		Rank-Acc	
	MSE Loss ↓	Mean L1 Loss ↓	Mean Prec.(%) ↑	MAE@2(%) ↑
Matrix Factorization	$2.16e^{-2}(1.19e^{-4})$	$9.47e^{-2}(2.89e^{-4})$	44.33(0.69)	83.16(0.73)
Neural Collaborative Filtering	$1.58e^{-2}(4.22e^{-5})$	$8.94e^{-2}(3.10e^{-4})$	41.76(1.22)	**84.98(0.42)**
+ Factor Enhanced	$\mathbf{1.25e^{-2}(3.35e^{-6})}$	$\mathbf{7.88e^{-2}(6.31e^{-5})}$	**45.45(0.33)**	84.54(0.27)
Only Factor	$1.75e^{-2}(2.07e^{-5})$	$8.57e^{-2}(1.48e^{-4})$	33.47(0.12)	84.08(0.37)

Table 1: Comparison of prediction methods for LLM performance. **Bold** indicates the best-performed.

- Further Improvement Through Model Development.

NCF > MF

- Increasing Accuracy by Incorporating Design Factors

NCF(Factor Enhanced) > NCF

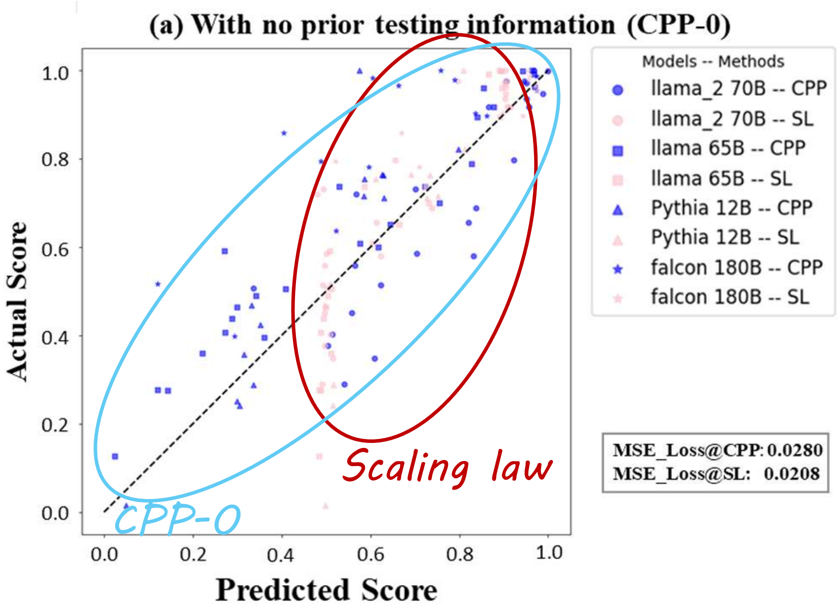
- Supporting Predictions based Solely on Factors.

Only Factor

Generalization for New Model

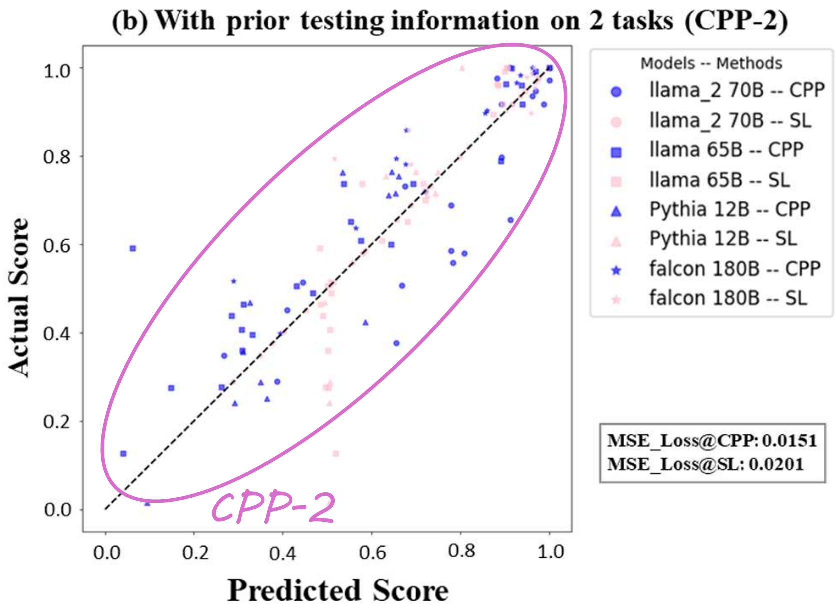
CPP-0 = predicting a model with no prior testing information.
CPP-2 = prediction a model with prior testing information on 2 tasks.

- CPP demonstrates greater adaptability than SL.



(a) With no prior testing information (CPP-0)

- CPP can utilize other tasks' performance to enhance prediction.



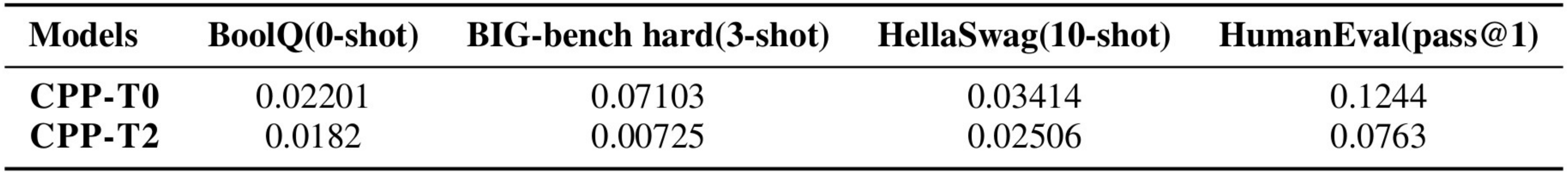
CPP-2 better than CPP-0

Dynamic Predictability = Iteration of ``evaluation'' and ``prediction''

evaluating simpler tasks can improve predictions for LLM performance on more complex tasks.

Generalization for New Task

CPP-T0 = predict performance on one task with no prior testing information;
CPP-T2 = predict performance on one task with prior testing information on 2 models.



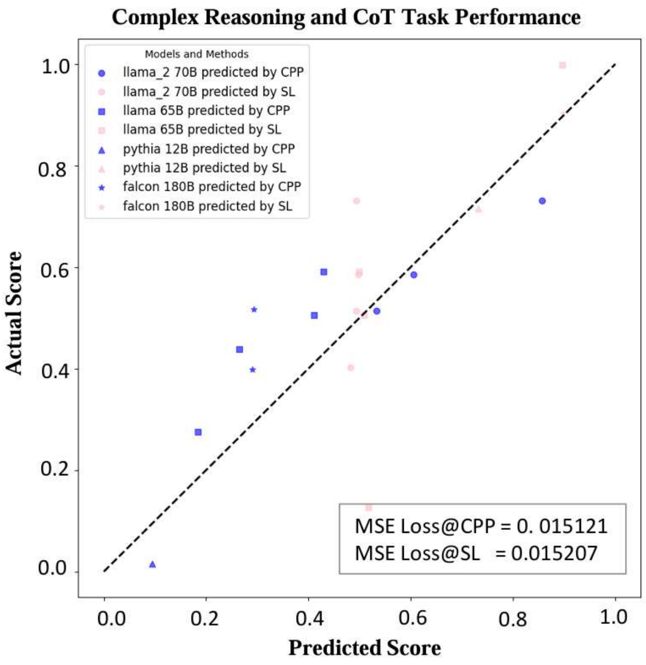
Models	BoolQ(0-shot)	BIG-bench hard(3-shot)	HellaSwag(10-shot)	HumanEval(pass@1)
CPP-T0	0.02201	0.07103	0.03414	0.1244
CPP-T2	0.0182	0.00725	0.02506	0.0763

Predicting Performance on Complex Reasoning Tasks

“Emergent” phenomena in Complex Reasoning Tasks: challenges associated with predicting performance from smaller models(7B) when the scale of a model expands significantly (70B), resulting in discontinuous leaps in model capabilities.

→ Difficult to predict

GSM8K, BBH, HUMANEVAL, MBPP

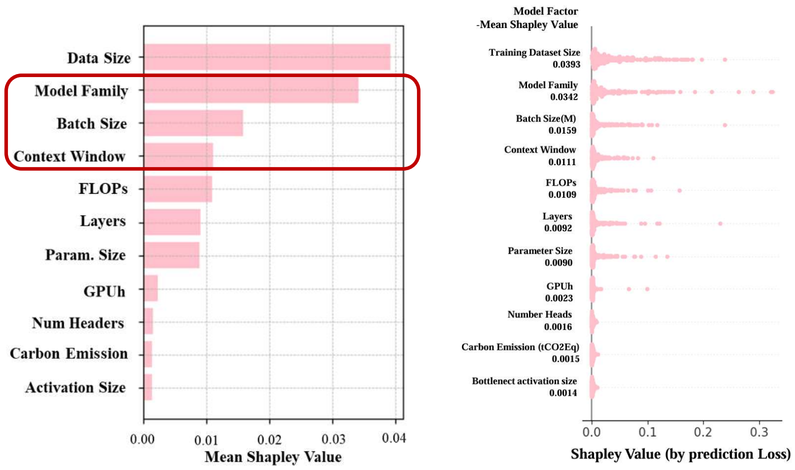


CPP better than SL

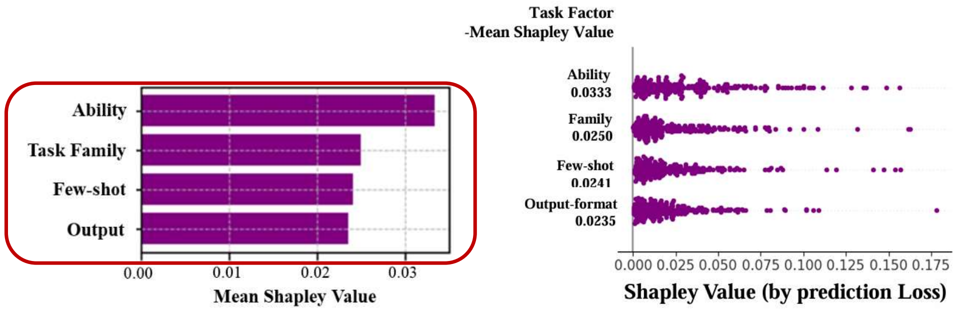
Factor Importance Analysis

CPP provides a base to analyze each design factor’s importance

The Shapley value, $\phi_i(v)$, quantifies the average marginal contribution of a factor i across all possible combinations of factors, and we utilize Shapley-Value for Factor Importance Analysis.



Model Factors



Task Factors

- Besides Data Size and Param.Size, other design factors significantly influence predictive outcomes.
- Task factors also have an important role in prediction.

Conclusion

- Predictability beyond Scaling Law

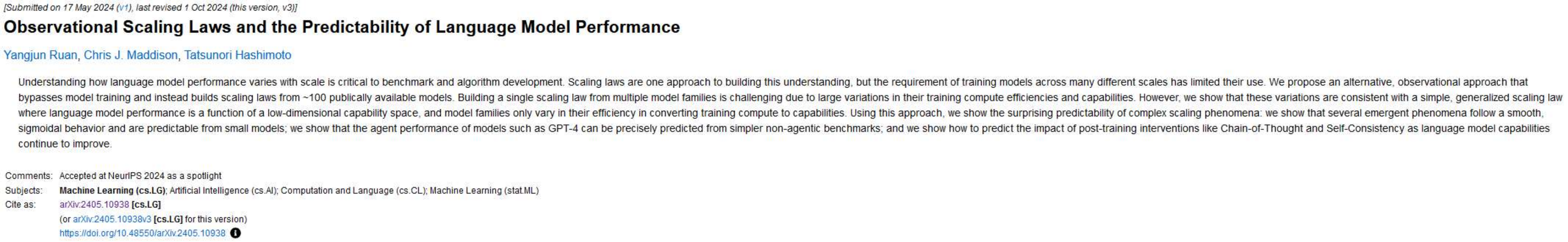
- Relationship Research among Models and Tasks-level

  We need collaborative research via open-source design factors

- Efficient Evaluation with Dynamic Predictability

  Predictability ⇄ Evaluation

Fun Facts



[Submitted on 17 May 2024 (v1), last revised 1 Oct 2024 (this version, v3)]

Observational Scaling Laws and the Predictability of Language Model Performance

Yangjun Ruan, Chris J. Maddison, Tatsunori Hashimoto

Understanding how language model performance varies with scale is critical to benchmark and algorithm development. Scaling laws are one approach to building this understanding, but the requirement of training models across many different scales has limited their use. We propose an alternative, observational approach that bypasses model training and instead builds scaling laws from ~100 publically available models. Building a single scaling law from multiple model families is challenging due to large variations in their training compute efficiencies and capabilities. However, we show that these variations are consistent with a simple, generalized scaling law where language model performance is a function of a low-dimensional capability space, and model families only vary in their efficiency in converting training compute to capabilities. Using this approach, we show the surprising predictability of complex scaling phenomena: we show that several emergent phenomena follow a smooth, sigmoidal behavior and are predictable from small models; we show that the agent performance of models such as GPT-4 can be precisely predicted from simpler non-agentic benchmarks; and we show how to predict the impact of post-training interventions like Chain-of-Thought and Self-Consistency as language model capabilities continue to improve.

Comments:	Accepted at NeurIPS 2024 as a spotlight
Subjects:	**Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI); Computation and Language (cs.CL); Machine Learning (stat.ML)
Cite as:	arXiv:2405.10938 **[cs.LG]** (or arXiv:2405.10938v3 **[cs.LG]** for this version) https://doi.org/10.48550/arXiv.2405.10938

Maybe we should aim higher and be more confident ☺

Thank you~