Large Language Models

Scaling Laws

ELL881 · AIL821

Sourish Dasgupta
Assistant Professor, DA-IICT, Gandhinagar
https://daiict.ac.in/faculty/sourish-dasgupta

2020 Arxiv (cs); Johns Hopkins University OpenAI

Kaplan Scaling Laws at a glance:

Power Law	Scale (tokenization-dependent)
$\alpha_N = 0.076$	$N_c = 8.8 \times 10^{13}$ params (non-embed)
$\alpha_D = 0.095$	$D_c = 5.4 \times 10^{13}$ tokens
$\alpha_C = 0.057$	$C_c = 1.6 \times 10^{7}$ PF-days
$\alpha_C^{\min} = 0.050$	$C_c^{\min} = 3.1 \times 10^{8}$ PF-days
$\alpha_B = 0.21$	$B_* = 2.1 \times 10^{8}$ tokens
$\alpha_S = 0.76$	$S_c = 2.1 \times 10^{3}$ steps

Parameters	Data	Compute	Batch Size	Equation
N	∞	∞	Fixed	$L(N) = (N_c/N)^{\alpha_N}$
∞	D	Early Stop	Fixed	$L(D) = (D_c/D)^{\alpha_D}$
Optimal	∞	C	Fixed	$L(C) = (C_c/C)^{\alpha_C}$ (naive)
N_{opt}	D_{opt}	$C_{\min}$	$B \ll B_{\text{crit}}$	$L(C_{\min}) = \left(C_c^{\min}/C_{\min}\right)^{\alpha_C^{\min}}$
N	D	Early Stop	Fixed	$L(N,D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$
N	∞	S steps	B	$L(N,S) = \left(\frac{N_c}{N}\right)^{\alpha_N} + \left(\frac{S_c}{S_{\min}(S,B)}\right)^{\alpha_S}$

Joint.

Is there any other alternative law?

Turns out there is!

Total params.

$$\mathrm{Loss}(N_T, D) = \frac{N_c}{N_T^\alpha} + \frac{D_c}{D^\beta} + E,$$

Lower bound

Loss that is intrinsic to the Dataset.

$$\mathrm{Loss}(N_T, C_T) = \frac{N_c}{N_T^\alpha} + \frac{D_c}{(C_T/6N_T)^\beta} + E$$

Chinchilla (Hoffman) Scaling Law

The Chinchilla (Hoffman) Scaling Law

$$\mathrm{Loss}(N_T, D) = \frac{N_c}{N_T^{\alpha}} + \frac{D_c}{D^{\beta}} + E \rightarrow L(N, D) = 1.69 + \frac{406.4}{N^{0.34}} + \frac{410.7}{D^{0.28}}$$

$$N_{opt}(C) = G(C/6)^{a} \quad D_{opt}(C) = G^{-1}(C/6)^{b}$$

$$\text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}} \quad a = \frac{\beta}{\alpha+\beta} \quad b = \frac{\alpha}{\alpha+\beta}$$

Fitting the constants, yields: $\alpha \approx \beta$ → best curve fitting

i.e. equal scaling of **N** and **D**.

Chinchilla Scaling Law vs. Kaplan Scaling Law

$$\text{Kaplan:} N^*_{\backslash E} \propto C_{\backslash E}^{0.73}$$

$$\text{Chinchilla:} N^*_T \propto C_T^{0.50}.$$

without Embed Matrix

Performance penalty is $N^{0.75}$ / D

- if model increases 8x, dataset must increase 5x

vs.

Fitting the constants, yields: $\alpha \approx \beta$

i.e. equal scaling of ***N*** and ***D***.

param for Embed Matrix.

$$N_T = N_E + N_{\backslash E},$$

$$N_E = (h + v)d,$$

context . vocab .

$$C_T = 6N_T D = 6(N_E + N_{\backslash E})D,$$

$$C_{\backslash E} = 6N_{\backslash E}D.$$

The (revised) Chinchilla Scaling Law

scaling factor of Dataset.

$$L(N, D) = 1.82 + \frac{514.0}{N^{0.35}} + \frac{2115.2}{D^{0.37}}$$

be better fit.

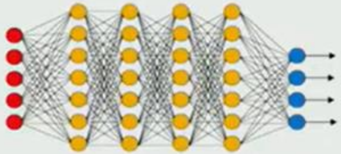
$$\boxed{L(N, D) = 1.69 + \frac{406.4}{N^{0.34}} + \frac{410.7}{D^{0.28}}}$$

→ Chinchilla law.

Is it a problem with our point-of-*view*?

LLMs “*seems*” to get more intelligent with the following:

Amount of training data

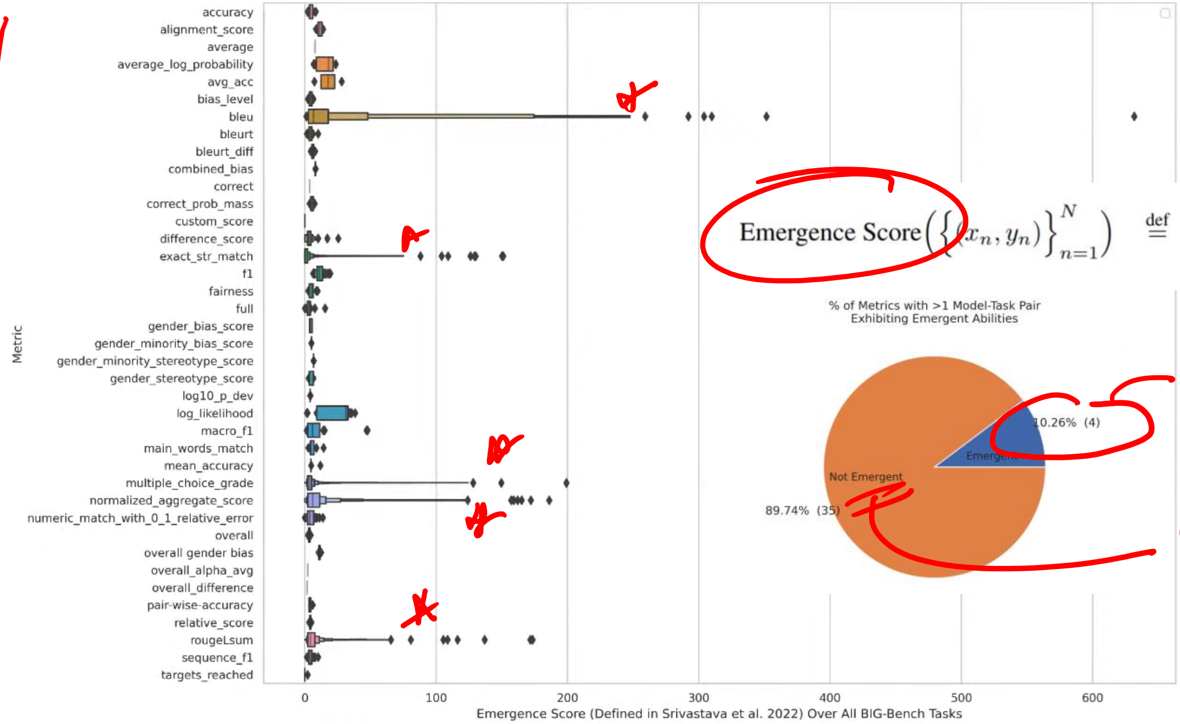


Model size (parameters)

$$\boxed{\mathcal{L}_{CE}(N) = \left(\frac{N}{c}\right)^{\alpha}}$$

Amount of compute (or time)

Motivation: Not all metrics score same (Emergence Score)



$$\text{Emergence Score}\left(\left\{(x_n, y_n)\right\}_{n=1}^{N}\right) \overset{\text{def}}{=} \frac{\text{sign}(\arg\max_i y_i - \arg\min_i y_i)(\max_i y_i - \min_i y_i)}{\sqrt{\text{Median}(\{(y_i - y_{i-1})^2\}_i)}}$$

10%

most measures

Is your accuracy metric non-linear or discontinuous?

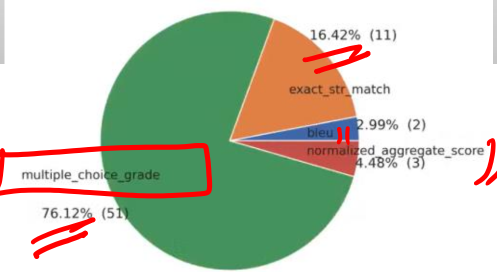
Benchmark.

> 92% of BIG-BENCH:

matters

$$\text{Multiple Choice Grade} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if highest probability mass on correct option} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Exact String Match} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if output string exactly matches target string} \\ 0 & \text{otherwise} \end{cases}$$

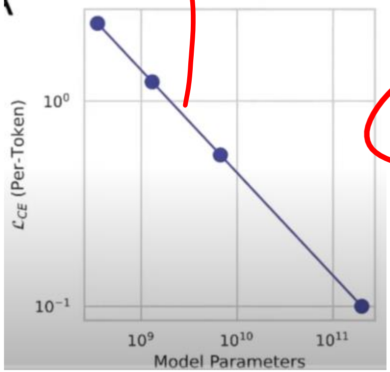


Too challenging for smaller models!
Is it really worth??

Power Law in play!

→ Kaplan.

Cross Entropy.

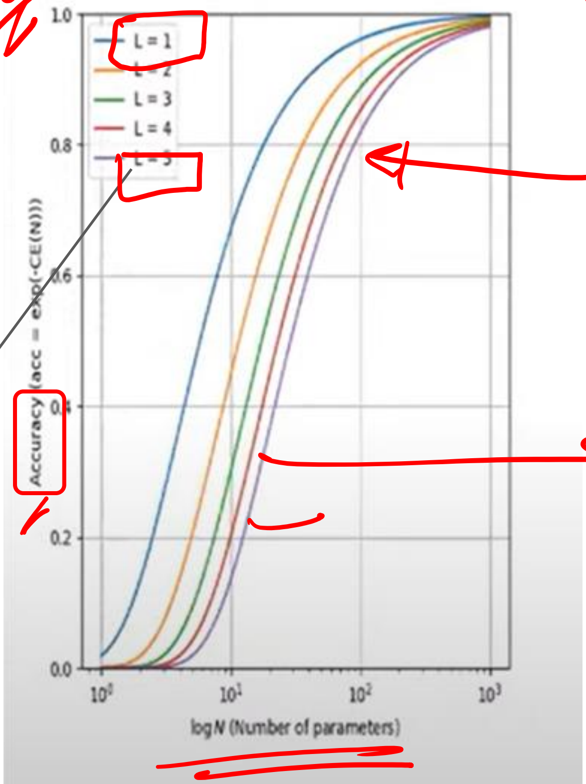


$$\mathcal{L}_{CE}(N) = \left(\frac{N}{c}\right)^{\alpha} = -\log \hat{p}_{v^*}(N)$$

$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^{\alpha}\right)$$

$$L(D_0, N) = B_0 + \frac{B_1}{N^{\alpha_W}} + \ldots$$

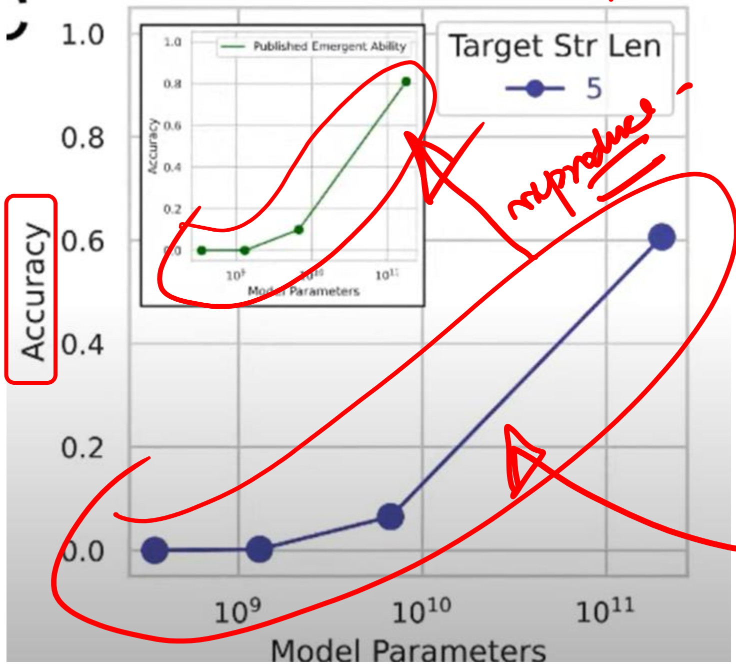
Length of token



→ S-curve.

Problem with Non-linear Measure: Eg.: *Exact string match*

Task: *Add k-digit integers*



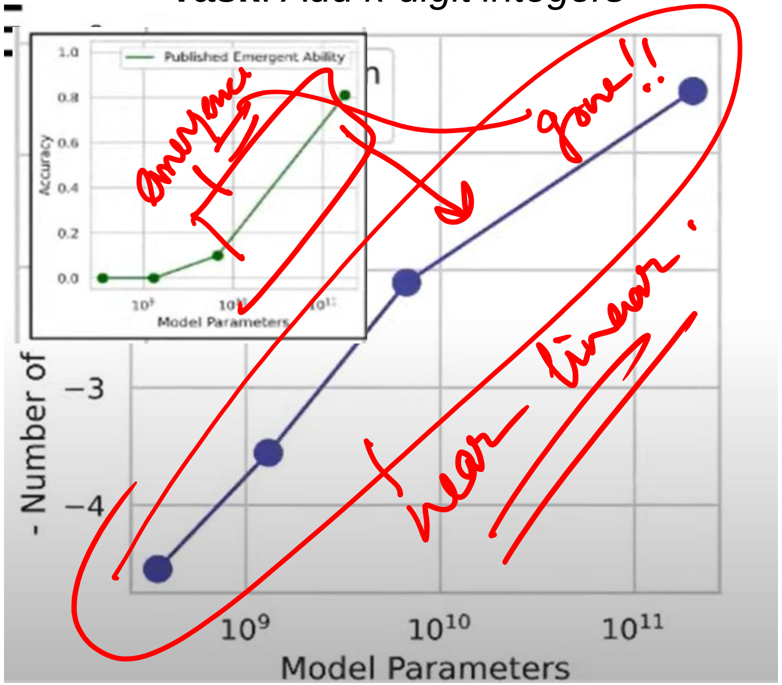
1	if all K+1 digits in model's output are correct
0	otherwise

$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^{\alpha} \right)$$

$$\text{Accuracy}(N) \approx p_N(\text{single token correct})^{\text{num. of tokens}}$$

Change of perspective: Measure: *Edit distance*

Task: *Add k-digit integers*



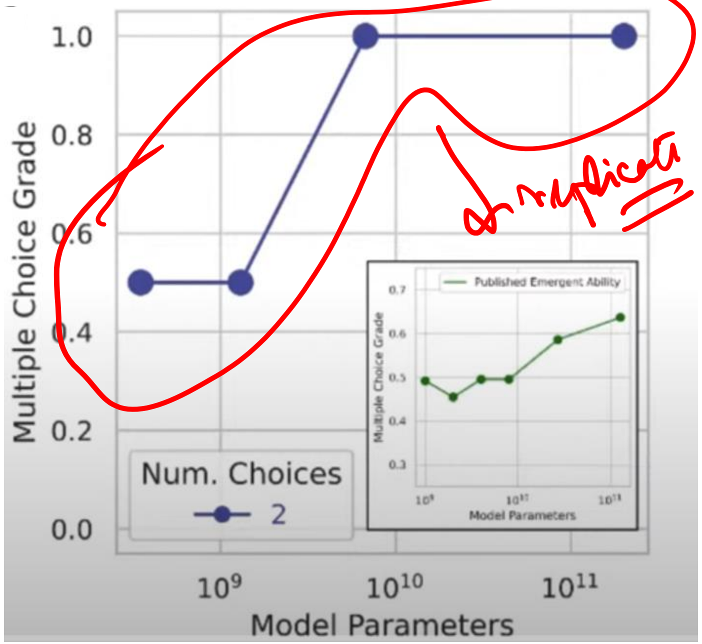
1	if all K+1 digits in model's output are correct
0	otherwise

$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^{\alpha}\right)$$

$$\text{Edit Distance}(N) \approx L\left(1 - p_N(\text{single token correct})\right)$$

Problem with Discontinuous Measure: Eg.: *MCG*

Task: *Choose one of two*

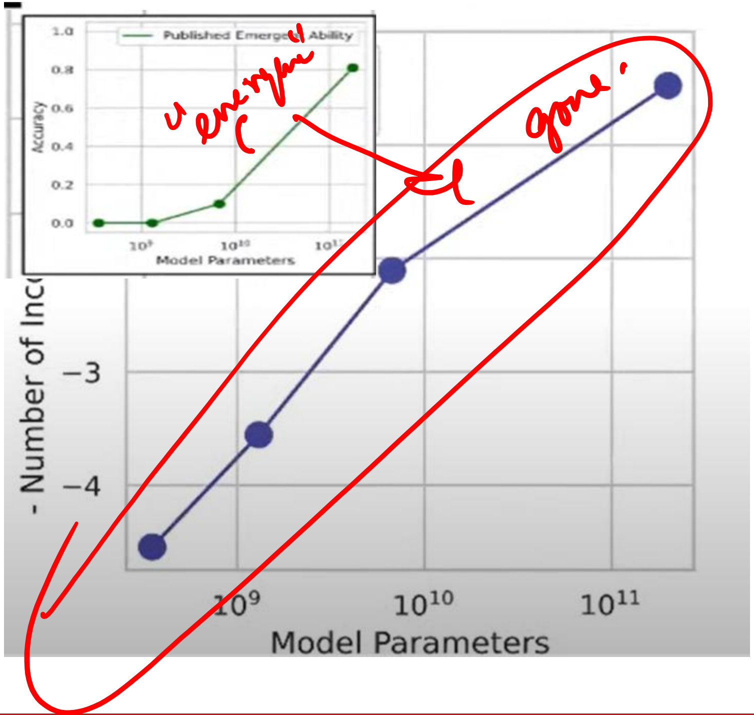


1	if highest probability mass on correct option
0	otherwise

$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^{\alpha} \right)$$

Change of perspective: Measure: *Brier Score*

Task: *Choose one of two*

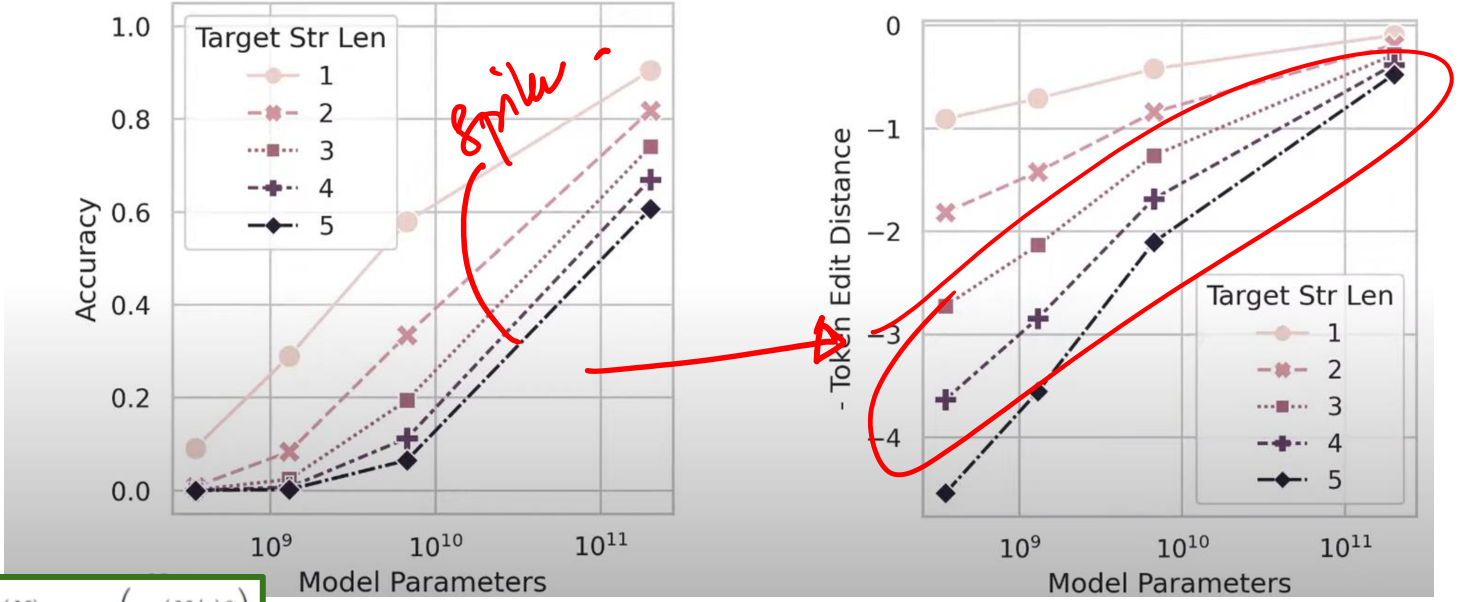


1	if all K+1 digits in model's output are correct
0	otherwise

$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^{\alpha} \right)$$

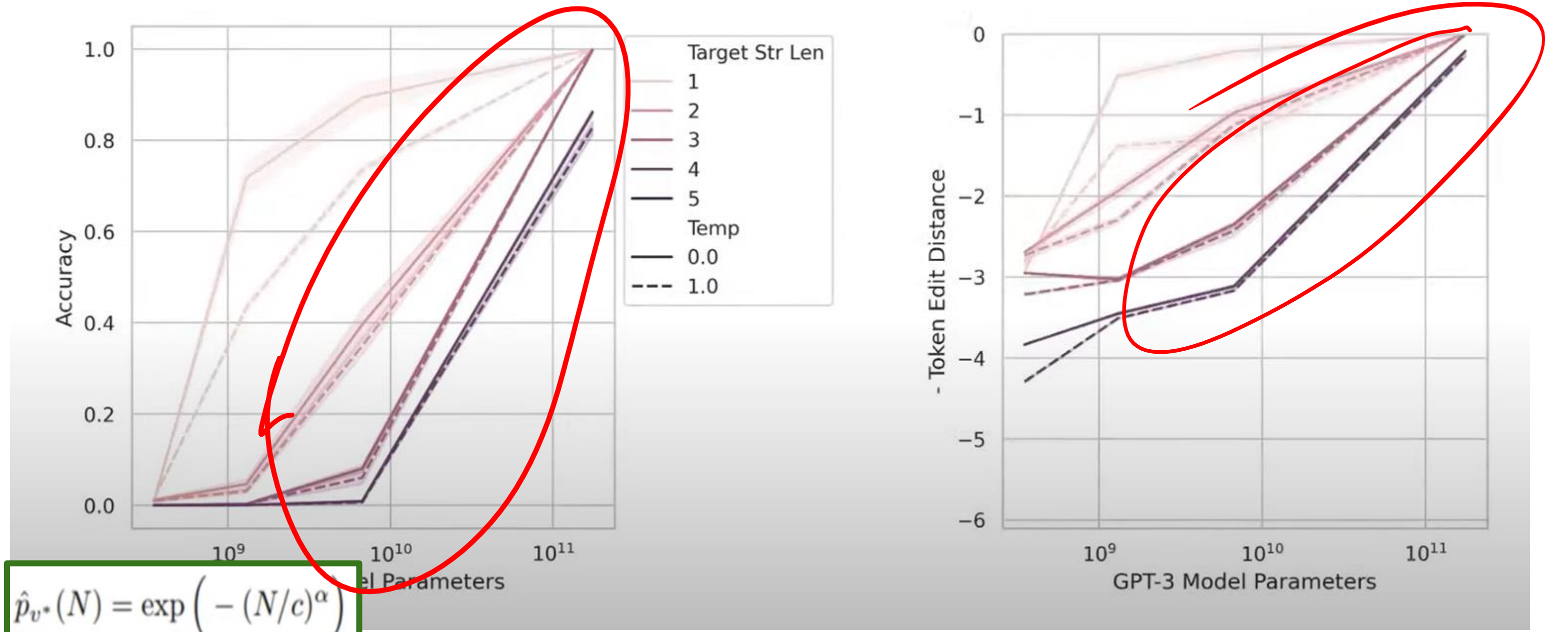
Brier Score = (1 – probability mass on correct option)^2

Prediction: Power Law vs. Near-Linear counterpart



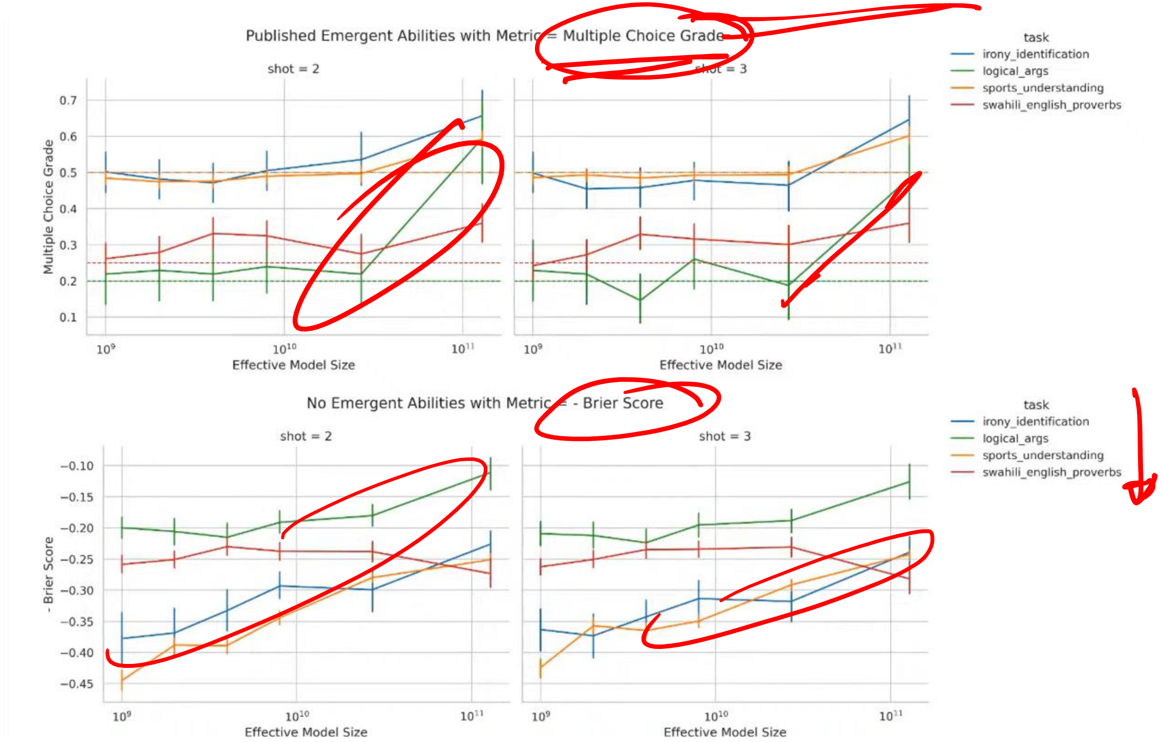
$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^\alpha\right)$$

Results on GPT3.5/3: Task: 2-digit integer multiplication



$$\hat{p}_{v^*}(N) = \exp\left(-(N/c)^\alpha\right)$$

Does the claim work for Google BIG-BENCH benchmark?



Key Takeaways

- Want to predict without the theatrics? Choose a *metric that's "soft"* (in the continuous sense)
- There's *no sudden jump* in reality ("*most*" can be predicted on a near-linear scale)
- Do we really need the power law of scale? Maybe not!