

# Emergent Abilities of Large Language Models

Jason Wei et al.

Transactions on Machine Learning Research (2022)

Summary by

Mohammed Aldosari and Rutuja Talekar

Tuesday Feb 21, 2023

# Outline

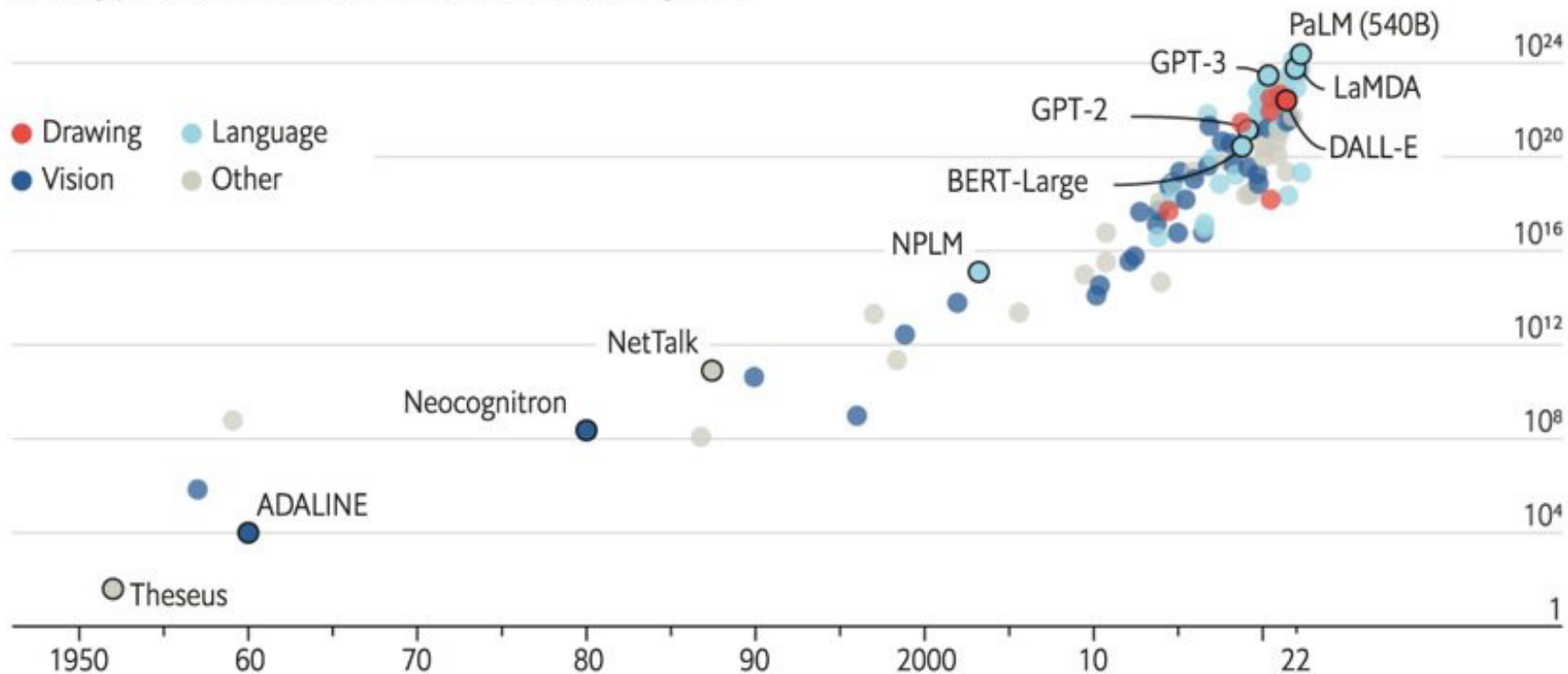
- Motivation
- Background
- Analyses
- Discussion
- Summary and Conclusion

# The blessings of scale

## The blessings of scale

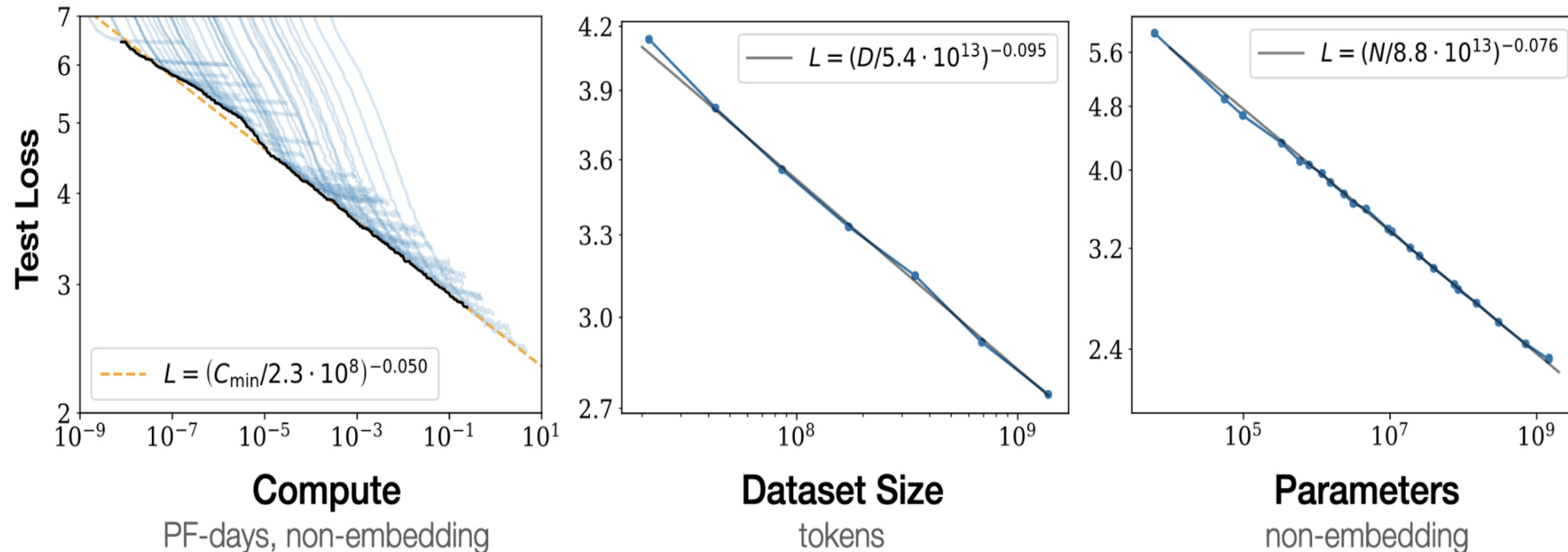
AI training runs, estimated computing resources used

Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

Language modeling performance improves smoothly as we increase the (1) model size, (2) data set size, and (3) amount of compute.



- “**Emergence** is when *quantitative* changes in a system result in *qualitative* changes in behavior.”
- “An ability is **emergent** if it is not present in smaller models but is present in larger models.”



## Future ML Systems Will Be Qualitatively Different

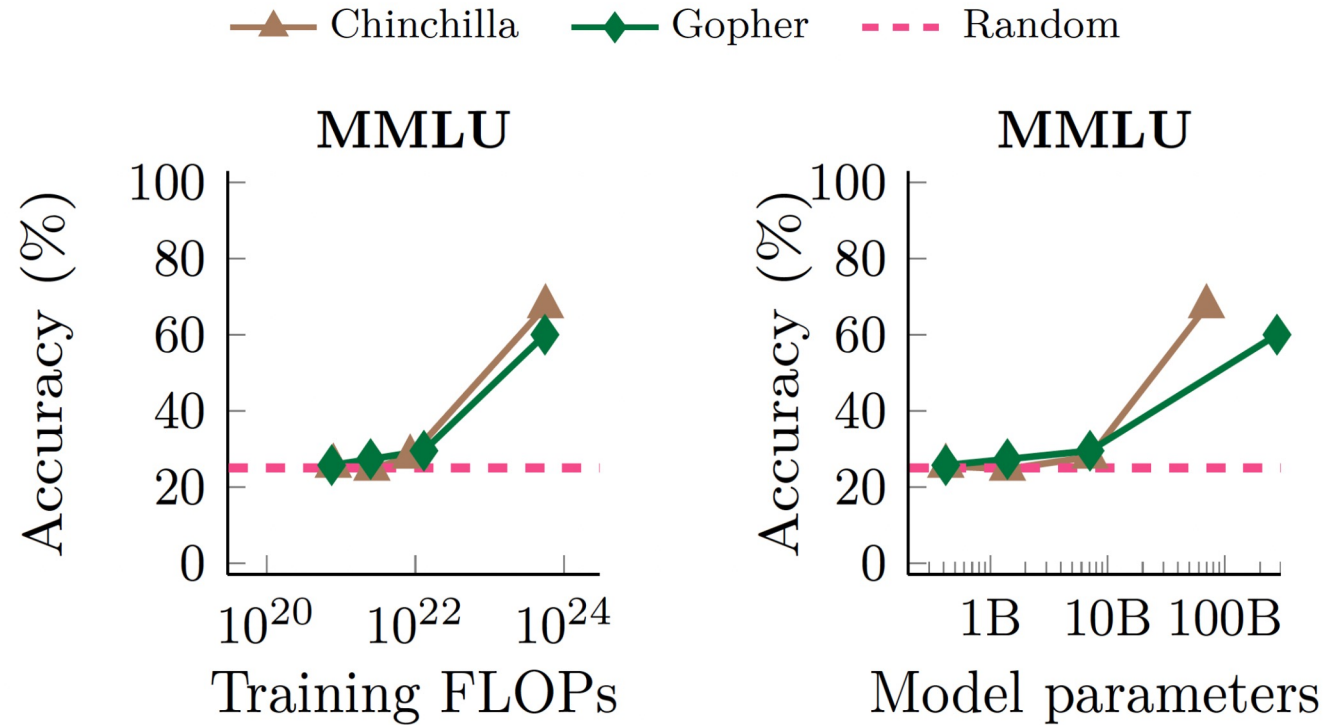
JAN 11, 2022 • 7 MIN READ

In 1972, the Nobel prize-winning physicist Philip Anderson wrote the essay "[More Is Different](#)". In it, he argues that quantitative changes can lead to qualitatively different and unexpected phenomena. While he focused on physics, one can find many examples of More is Different in other domains as well, including biology, economics, and computer science. Some examples of More is Different include:

- **Uranium.** With a bit of uranium, nothing special happens; with a large amount of uranium packed densely enough, you get a nuclear reaction.
- **DNA.** Given only small molecules such as calcium, you can't meaningfully encode useful information; given larger molecules such as DNA, you can encode a genome.
- **Water.** Individual water molecules aren't wet. Wetness only occurs due to the interaction forces between many water molecules interspersed throughout a fabric (or other material).
- **Traffic.** A few cars on the road are fine, but with too many you get a traffic jam. It could be that 10,000 cars could traverse a highway easily in 15 minutes, but 20,000 on the road at once could take over an hour.
- **Specialization.** Historically, in small populations, virtually everyone needed to farm or hunt to survive; in contrast, in larger and denser communities, enough food is produced for large fractions of the population to specialize in non-agricultural work.

While some of the examples, like uranium, correspond to a sharp transition, others like specialization are more continuous. I'll use **emergence** to refer to qualitative changes that arise from quantitative increases in scale, and **phase transitions** for cases where the change is sharp.

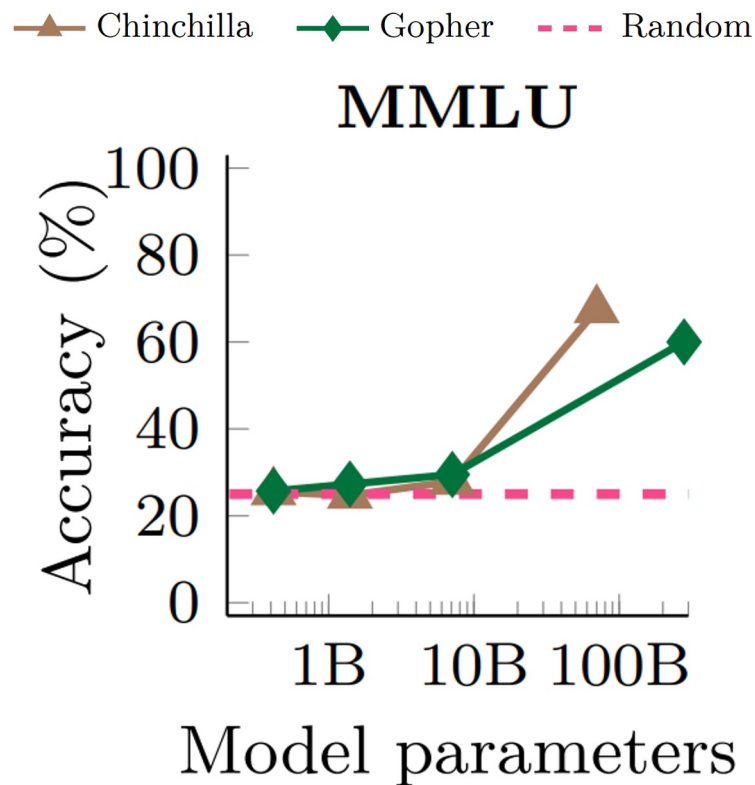
# Model performance depends most strongly on scale



## Massive Multi-Task Language Understanding Benchmark (MMLU)

57 tasks including elementary mathematics, US history, computer science, law, and more.

Emergent abilities also depend on other factors: not being limited by the amount of data, its quality, or the way we train the models.



Model	Parameters	Train tokens	Train FLOPs
Gopher	417M	300B	7.51E+20
	1.4B	300B	2.52E+21
	7.1B	300B	1.28E+22
	280B	325B	5.46E+23
Chinchilla	417M	314B	7.86E+20
	1.4B	314B	2.63E+21
	7.1B	[sic] 199B	8.47E+21
	70B	1.34T	5.63E+23

# Floating Point Operations (FLOPs)

- A rough measure of how expensive an algorithm or a *model* can be
- Denote number of additions, subtractions, multiplications or divisions of floating-point numbers
- Given vector  $a, b \in \mathbb{R}^n$ :  
Addition  $a + b$  requires  $n$  flops for  $n$  element-wise additions



# Example

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.linear = nn.Linear(2, 4, bias=False)

    def forward(self, x):
        x = self.linear(x)
        x = F.relu(x)
        return x

model = Model()

x = torch.randn(1, 2)

y = model(x)
```

# FLOPs

$$\begin{array}{ccc} \begin{array}{c} 1 \times 2 \\ \text{Input} \end{array} & \begin{array}{c} 2 \times 4 \\ \text{Trainable weights} \end{array} & \begin{array}{c} 1 \times 4 \\ \text{Output} \end{array} \\ [0.1 & 0.1] & \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} = [0.02 & 0.02 & 0.02 & 0.02] \end{array}$$

$$\begin{array}{ccccccc} \begin{array}{c} 3 \text{ FLOPs} \\ \text{---} \end{array} & \begin{array}{c} 3 \text{ FLOPs} \\ \text{---} \end{array} & \begin{array}{c} 3 \text{ FLOPs} \\ \text{---} \end{array} & \begin{array}{c} 3 \text{ FLOPs} \\ \text{---} \end{array} & \begin{array}{c} 4 \text{ FLOPs} \\ \text{---} \end{array} \\ [(0.1*0.2)+(0.1*0.2), & (0.1*0.2)+(0.1*0.2), & (0.1*0.2)+(0.1*0.2), & (0.1*0.2)+(0.1*0.2)] & \text{ReLU} \end{array}$$

**12 + 4 = 16 FLOPs for Forward Pass**  
**32 FLOPs for Backward Pass**

# Model Parameters

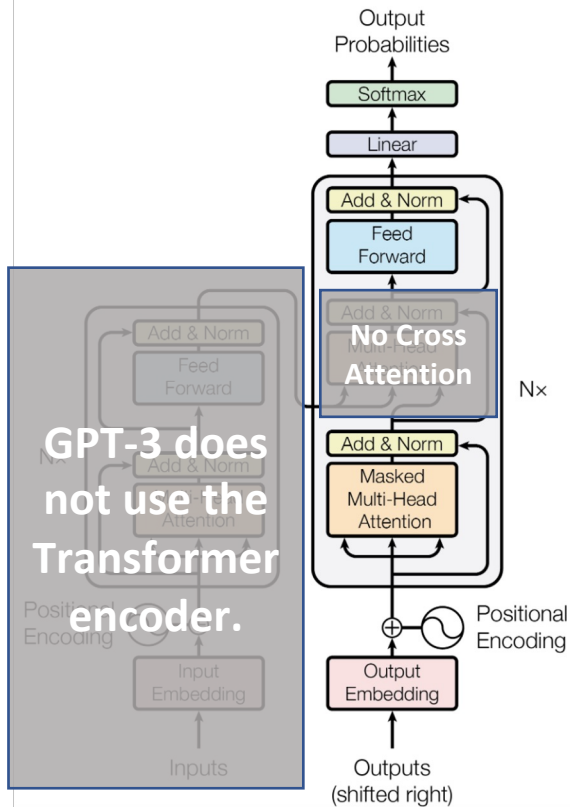
- Denote the total number of trainable parameters or weights of the model that can be estimated from the data.
- Weights are initialized randomly from a Gaussian or uniform distribution using different initialization strategies.

$$\begin{array}{ccc} \begin{array}{c} 1 \times 2 \\ \text{Input} \end{array} & \begin{array}{c} 2 \times 4 \\ \text{Trainable weights} \end{array} & \begin{array}{c} 1 \times 4 \\ \text{Output} \end{array} \\ [0.1 \quad 0.1] \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} & = & [0.02 \quad 0.02 \quad 0.02 \quad 0.02] \end{array}$$

8 Trainable parameters or weights

# GPT-3 Small

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$



layer norm weight + bias =  $[768 * 4]$   
**3072 parameters**

query + bias =  $[768, 12*64, 768]$   
**590592 parameters**

key + bias =  $[768, 12*64, 768]$   
**590592 parameters**

value + bias =  $[768, 12*64, 768]$   
**590592 parameters**

ffnn 1 + bias =  $[768, 768, 768]$   
**590592 parameters**

ffnn 2 + bias =  $[768, 768 * 4, 768]$   
**2360064 parameters**

ffnn 2 + bias =  $[768 * 4, 768, 768]$   
**2360064 parameters**

positional encodings =  $[1024, 768]$   
**786432 parameters**

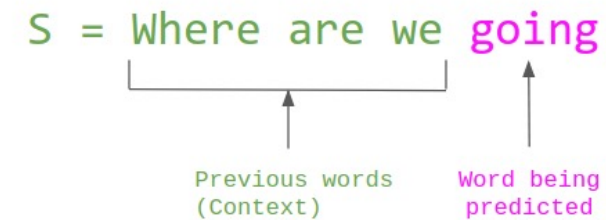
embeddings =  $[50257, 768]$   
**38597376 parameters**

**Total parameters =**  
 **$38597376 + 786432 +$**   
 **$12 * (3*(590592) + 590592 +$**   
 **$2*(2360064) + (3072))$**   
**= 124M**

**\* 12**

# Pre-training

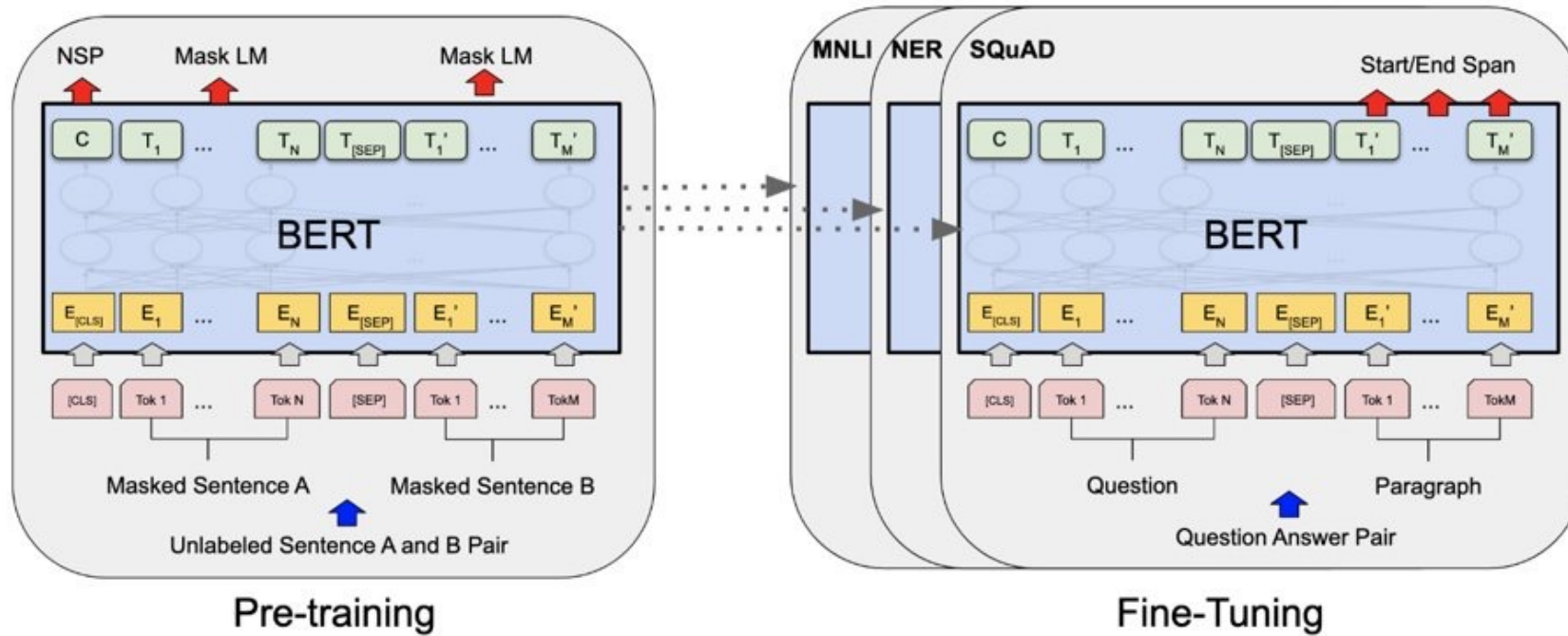
- The main objective of a pre-trained LM consists of some sort of objective predicting the probability of text  $x$ .
- To pre-train a language model, there are a couple of considerations:
  - **training objectives**
  - **noising functions**
  - **directionality of representations**



$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

# Fine-tuning

- Promptless Fine-tuning

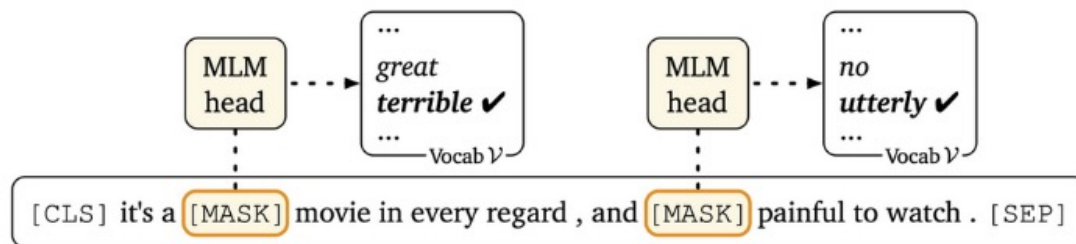


# Fine-tuning

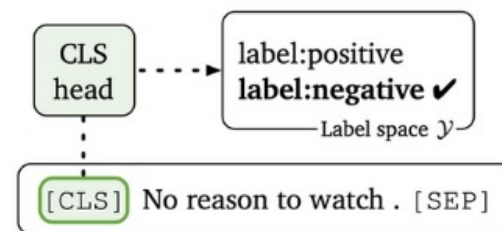
- Prompt Fine-tuning

Name	Notation	Example	Description
<i>Input</i>	$\mathbf{x}$	I love this movie.	One or multiple texts
<i>Output</i>	$\mathbf{y}$	++ (very positive)	Output label or text
<i>Prompting Function</i>	$f_{\text{prompt}}(\mathbf{x})$	[X] Overall, it was a [Z] movie.	A function that converts the input into a specific form by inserting the input $\mathbf{x}$ and adding a slot [Z] where answer $\mathbf{z}$ may be filled later.
<i>Prompt</i>	$\mathbf{x}'$	I love this movie. Overall, it was a [Z] movie.	A text where [X] is instantiated by input $\mathbf{x}$ but answer slot [Z] is not.
<i>Filled Prompt</i>	$f_{\text{fill}}(\mathbf{x}', \mathbf{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot [Z] is filled with any answer.
<i>Answered Prompt</i>	$f_{\text{fill}}(\mathbf{x}', \mathbf{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot [Z] is filled with a true answer.
<i>Answer</i>	$\mathbf{z}$	“good”, “fantastic”, “boring”	A token, phrase, or sentence that fills [Z]

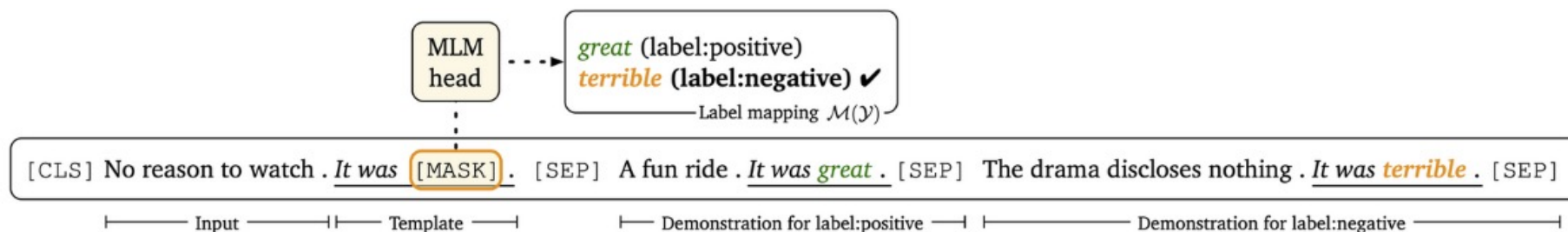
# Why Prompts?



(a) MLM pre-training



(b) Fine-tuning

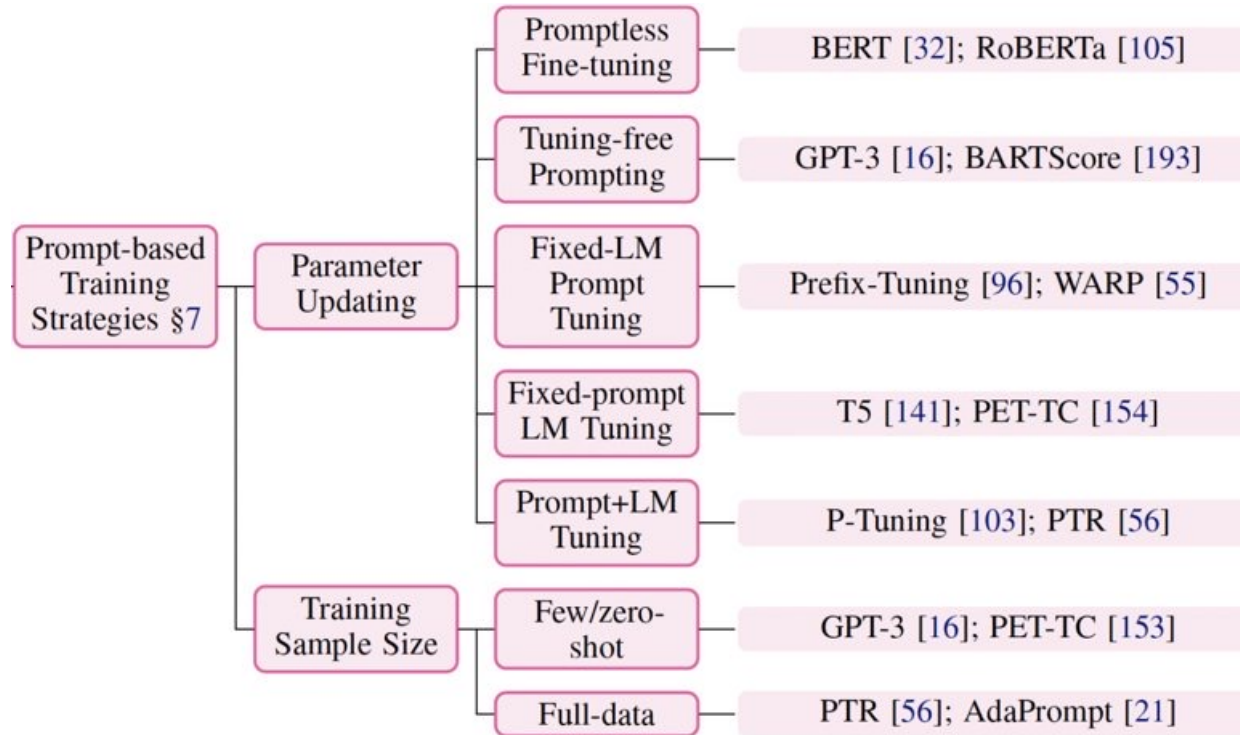


(c) Prompt-based fine-tuning with demonstrations (our approach)



# Fine-tuning

- Prompt Fine-tuning



# Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



# One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

1    Translate English to French:    ← *task description*

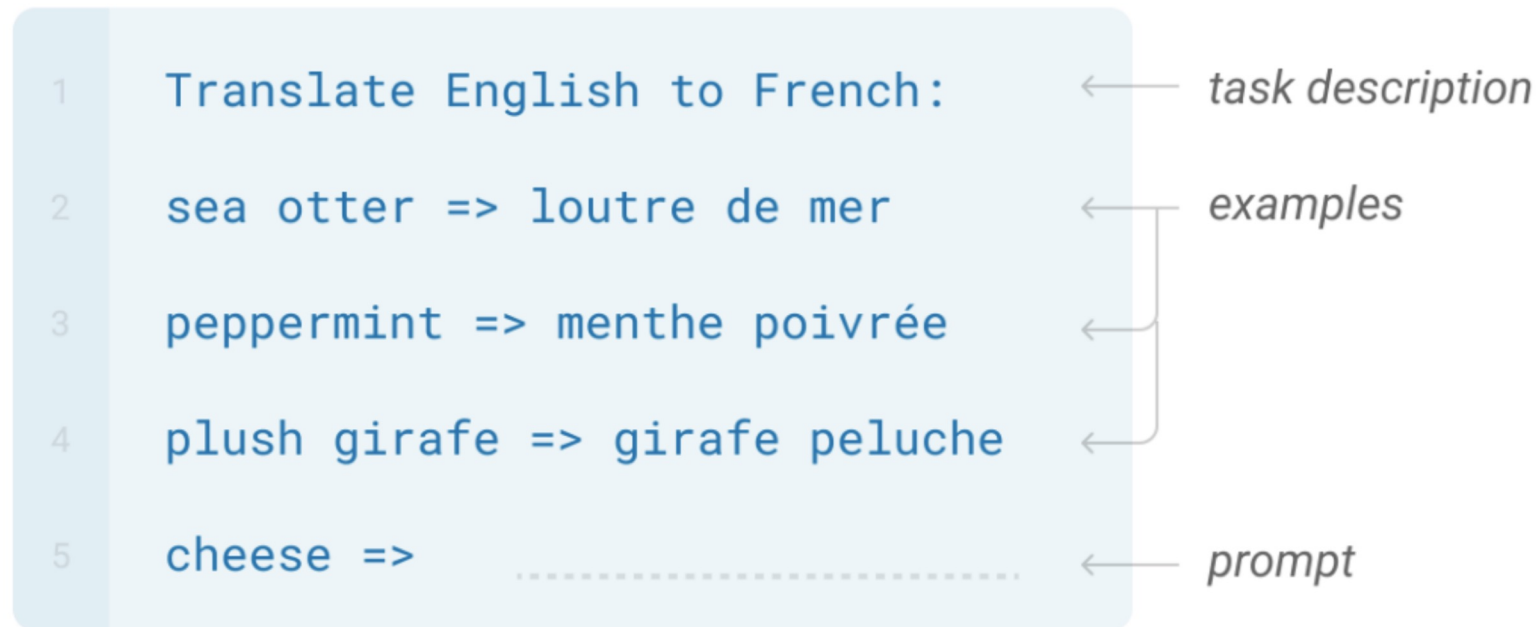
2    sea otter => loutre de mer    ← *example*

3    cheese =>    ← *prompt*

.....

# Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



The diagram shows a prompt structure for a few-shot learning task. It consists of five lines of text, each preceded by a number in a light blue box. The text is as follows:

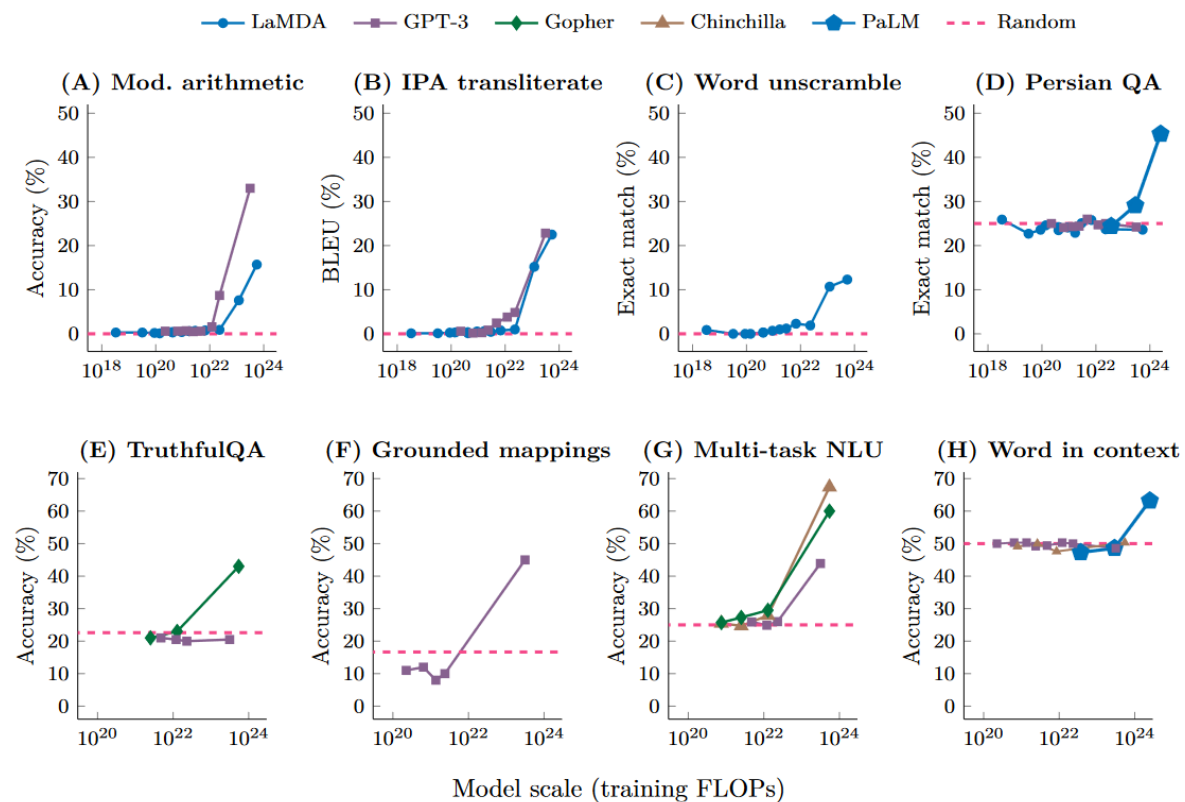
- 1 Translate English to French:
- 2 sea otter => loutre de mer
- 3 peppermint => menthe poivrée
- 4 plush girafe => girafe peluche
- 5 cheese => .....

Annotations with arrows point to specific parts of the prompt:

- An arrow labeled *task description* points to line 1.
- An arrow labeled *examples* points to lines 2, 3, and 4.
- An arrow labeled *prompt* points to line 5.

# Few-Shot Prompting Tasks

- Big-Bench.
- TruthfulQA.
- Grounded conceptual mappings.
- Multi-task language understanding (MMLU).
- Word in context.



# Big Bench

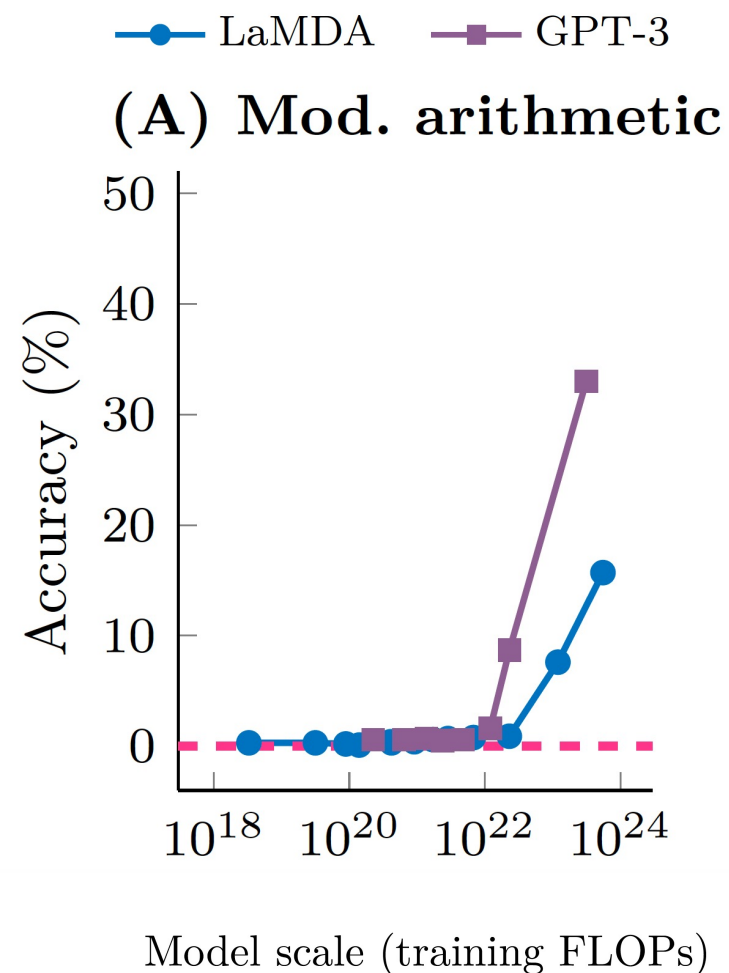
- Modified Arithmetic

Prompt:

```
In the following lines, the symbol -> represents a simple mathematical operation.  
100 + 200 -> 301  
838 + 520 -> 1359  
343 + 128 -> 472  
647 + 471 -> 1119  
64 + 138 -> 203  
498 + 592 ->
```

Answer:

1091



# Massive Multi-Task Language Understanding Benchmark (MMLU)

## Philosophy Test

1. A moral theory explains \_\_\_\_.

- A) why an action is right or wrong
- B) why one moral event caused another
- C) where a moral agent got her values
- D) why people do what they do

Answer: A

## Professional Accounting Test

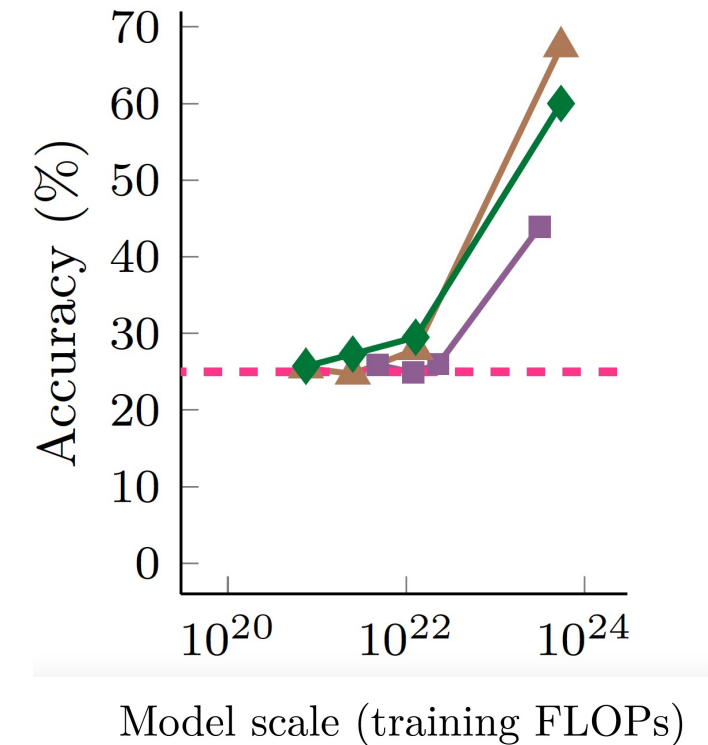
What is the price of a three-year bond (face value \$100), paying 5% coupons, with a yield of 6%?

- A) \$100
- B) \$104.29
- C) \$96.71
- D) \$97.33

Answer: D

—■— GPT-3    —◆— Gopher    —▲— Chinchilla    - - - Random

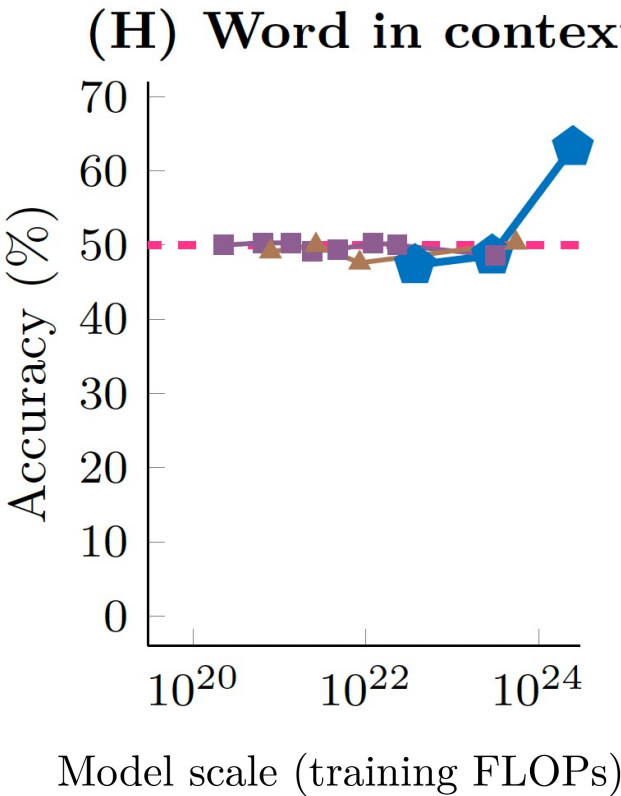
(G) Multi-task NLU



# Word-in-Context (WiC)

Label	Target	Context-1	Context-2
F	bed	There's a lot of trash on the <u>bed</u> of the river	I keep a glass of water next to my <u>bed</u> when I sleep
F	land	The pilot managed to <u>land</u> the airplane safely	The enemy <u>landed</u> several of our aircrafts
F	justify	<u>Justify</u> the margins	The end <u>justifies</u> the means
T	beat	We <u>beat</u> the competition	Agassi <u>beat</u> Becker in the tennis championship
T	air	<u>Air</u> pollution	Open a window and let in some <u>air</u>
T	window	The expanded <u>window</u> will give us time to catch the thieves	You have a two-hour <u>window</u> of clear weather to finish working on the lawn

—■— GPT-3    —▲— Chinchilla    —◆— PaLM    - - - Random





# Prompting Demo

- [Big bench - Fact Checking - Covid19 Scientific](#)
- [OpenAI Playground](#)

# Augmented Prompting Strategies

- Multi-step reasoning
- Instruction following
- Program execution
- Model calibration

*"A prompting technique is emergent if it hurts performance (compared to baseline) for small models, And improves baseline for large models"*



# Multi-Step Reasoning

## Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

## Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# Instruction Following

Fine-tuning on a mixture of tasks phrased as instructions

# Instruction Following

Fine-tuning on a mixture of tasks phrased as instructions

# Program Execution

Fine-tuning the models to predict intermediate outputs as scratchpad

## DIRECT EXECUTION PREDICTION

Consider the following Python function:

```
def remove_Occ(s,ch):
    for i in range(len(s)):
        if (s[i] == ch):
            s = s[0 : i] + s[i + 1:]
            break
    for i in range(len(s) - 1,-1,-1):
        if (s[i] == ch):
            s = s[0 : i] + s[i + 1:]
            break
    return s
```

Fill in the ??? below:  
assert remove\_Occ("PHP","P") == ???

Large Language Model

assert remove\_Occ("PHP", "P") == "H"

## SCRATCHPAD TRACING

Consider the following Python function:

```
def remove_Occ(s,ch):
    for i in range(len(s)):
        if (s[i] == ch):
            s = s[0 : i] + s[i + 1:]
            break
    for i in range(len(s) - 1,-1,-1):
        if (s[i] == ch):
            s = s[0 : i] + s[i + 1:]
            break
    return s
```

output = remove\_Occ("PHP","P")

What is the execution trace?

Large Language Model

```
state: {}
line: def remove_Occ(s,ch):
state: {"remove_Occ": "<callable_object remove_Occ>"}
line: output = remove_Occ("PHP","P")
state: {"ch": "P", "s": "PHP"}
line:     for i in range(len(s)):
state: {"ch": "P", "s": "PHP", "i": 0}
line:         if (s[i] == ch):
state: {"ch": "P", "s": "PHP", "i": 0}
line:             s = s[0 : i] + s[i + 1:]
state: {"ch": "P", "s": "HP", "i": 0}
line:             break
state: {"ch": "P", "s": "HP", "i": 0}
line:     for i in range(len(s) - 1,-1,-1):
state: {"ch": "P", "s": "HP", "i": 1}
line:         if (s[i] == ch):
state: {"ch": "P", "s": "HP", "i": 1}
line:             s = s[0 : i] + s[i + 1:]
state: {"ch": "P", "s": "H", "i": 1}
line:             break
state: {"ch": "P", "s": "H", "i": 1}
line:     return s
state: {"remove_Occ": "<callable_object remove_Occ>",
      "output": "H"}
```

# Instruction Following

Fine-tuning on a mixture of tasks phrased as instructions

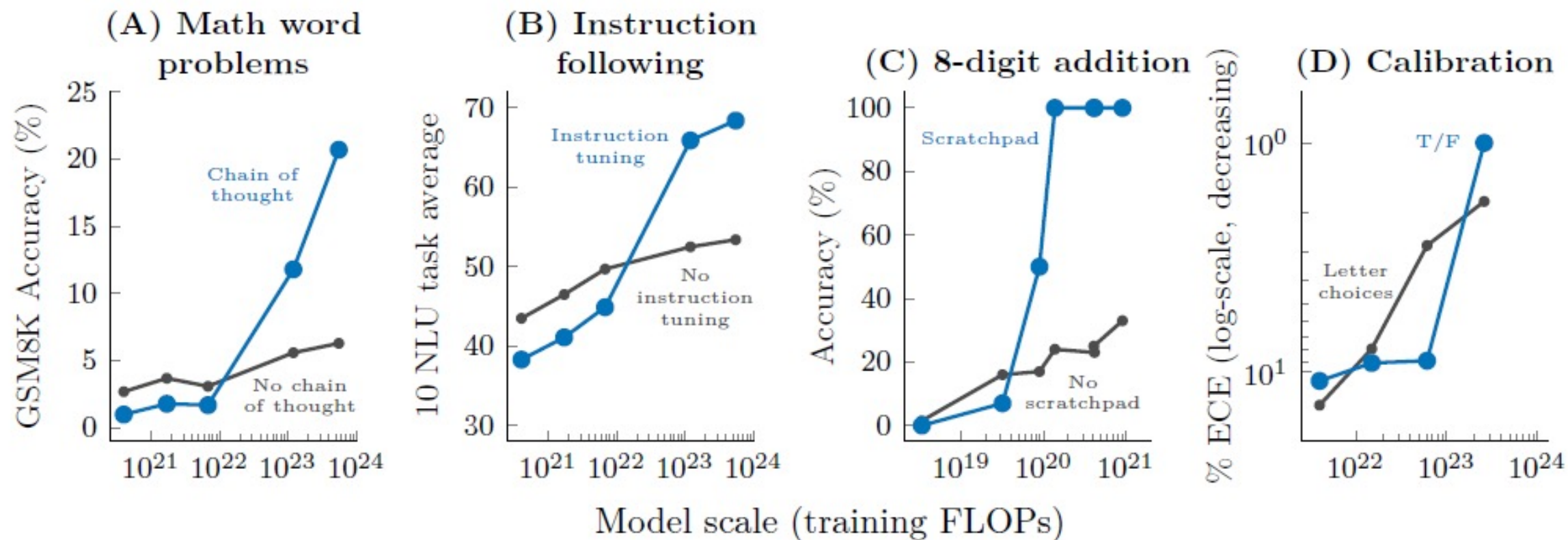
# Program Execution

Fine-tuning the models to predict intermediate outputs as scratchpad

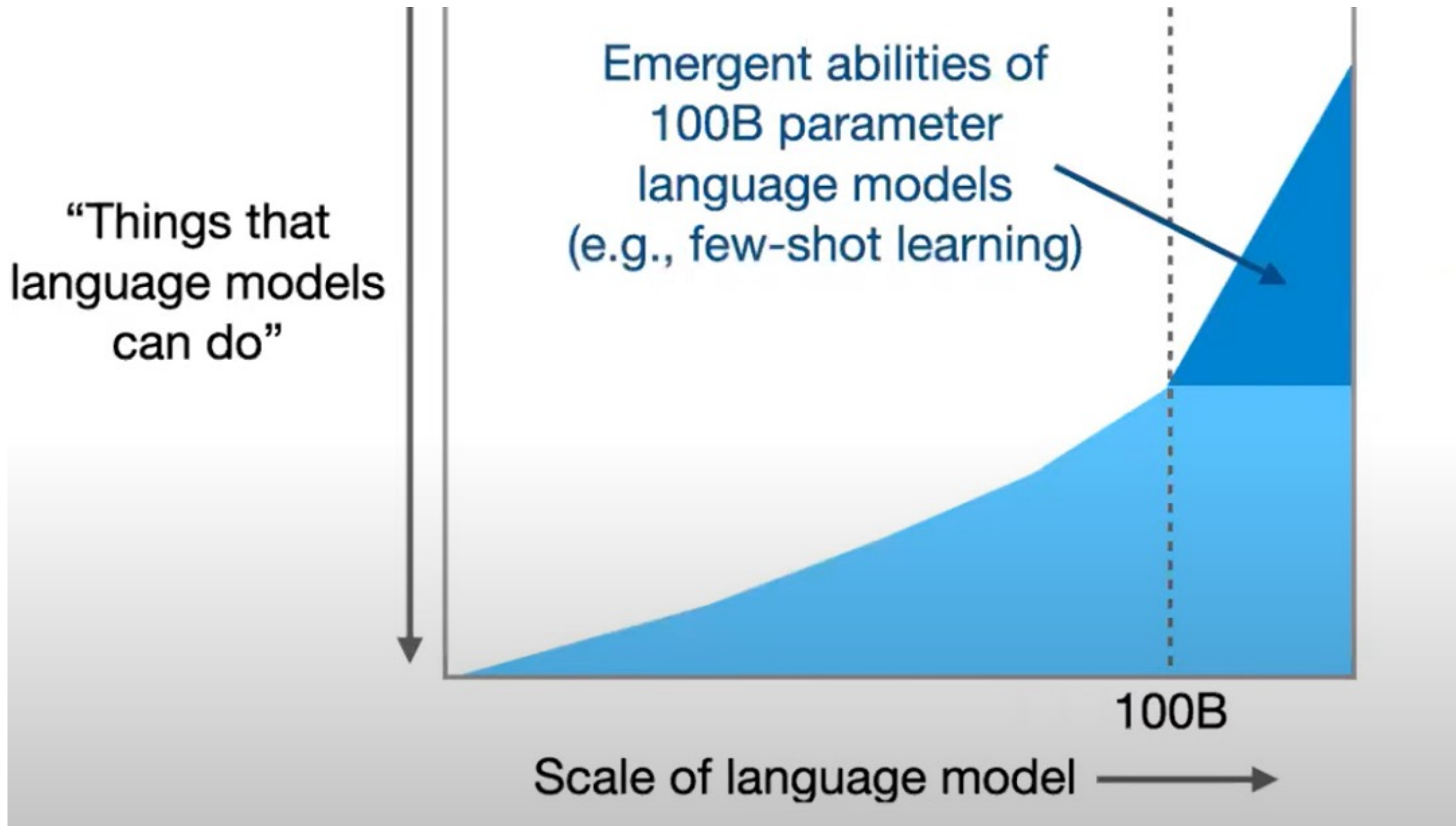
# Model Calibration

Predicting which questions the models will be able to answer correctly

# Emergence of Augmented Strategies









# Discussion

- Observation so far - Few shot prompting setup works with large language models
- "Emergent few shot prompted tasks are unpredictable"



# Discussion

- Observation so far - Few shot prompting setup works with large language models
- "Emergent few shot prompted tasks are unpredictable"
- **Explanations of emergence?**

# Evaluation Metrics for Emergence

- Using different metric can lead to ignorance of incremental improvements/ different scaling curve
- Example – Using exact match on long sequence targets
- Example – Not giving partial credits to multi-step problem

# Evaluation Metrics for Emergence

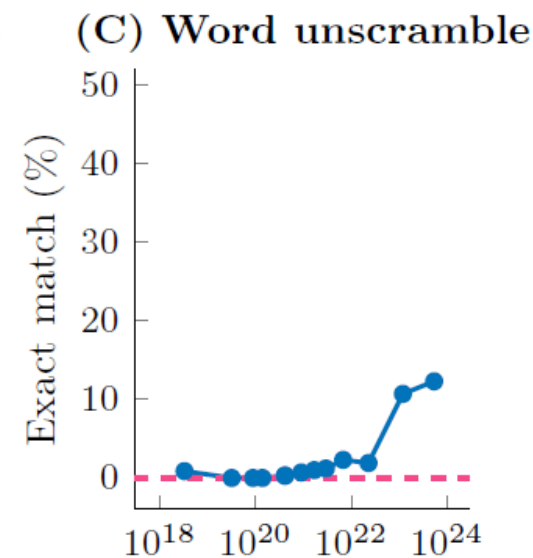
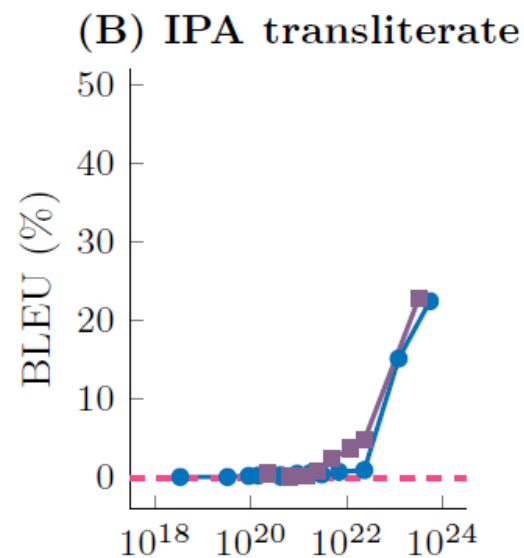
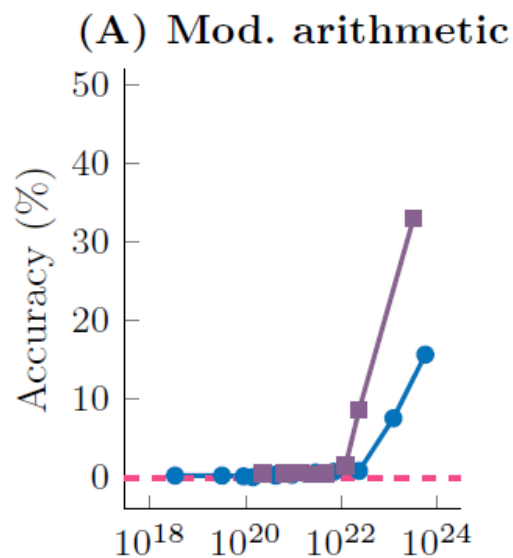
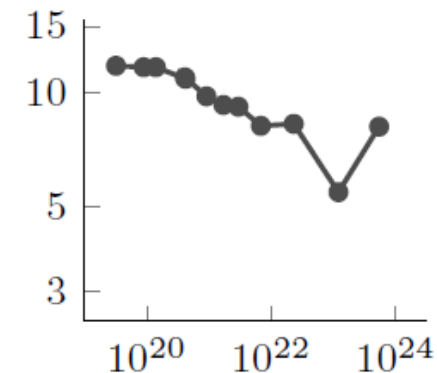
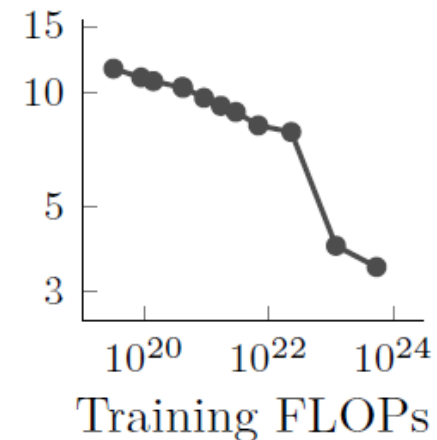
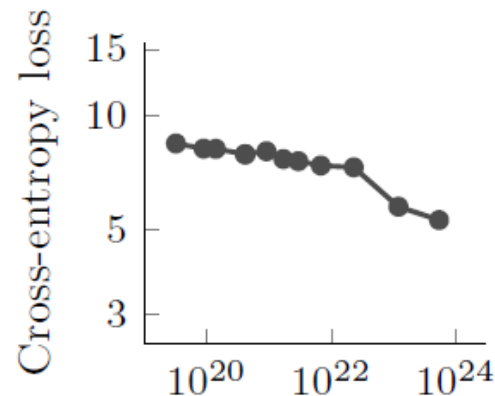
- Using different metric can lead to ignorance of incremental improvements
- Example – Using exact match on long sequence targets
- Example – Not giving partial credits to multi-step problem
- **Cross Entropy loss as a measure**
- Loss improves for even for smaller scale models over the close to random scale by accuracy, BLEU and exact match

# Evaluation Metrics for Emergence

## EM/BLEU/acc Vs Cross Entropy Loss

- Outcome 1: For the model scales where EM/BLEU/acc is random, cross-entropy loss also does not improve as scale increases. This outcome implies that for these scales, the model truly does not get any better at the tasks.
- Outcome 2: For the model scales where EM/BLEU/acc is random, cross-entropy loss does improve. This outcome implies that the models do get better at the task, but these improvements are not reflected in the downstream metric of interest. The broader implication is that scaling small models improves the models in a way that is not reflected in EM/BLEU/Acc, and that there is some critical model scale where these improvements enable the downstream metric to increase to above random as an emergent ability.

# Evaluation Metrics for Emergence



—●— LaMDA —■— GPT-3 —◆— Gopher —▲— Chinchilla —◆— PaLM - - - Random

# Emergence: Better Data

- Model scale is not the singular factor for unlocking an emergent ability
- Example - LaMDA 137B and GPT-3 175B lose to PaLM 62B

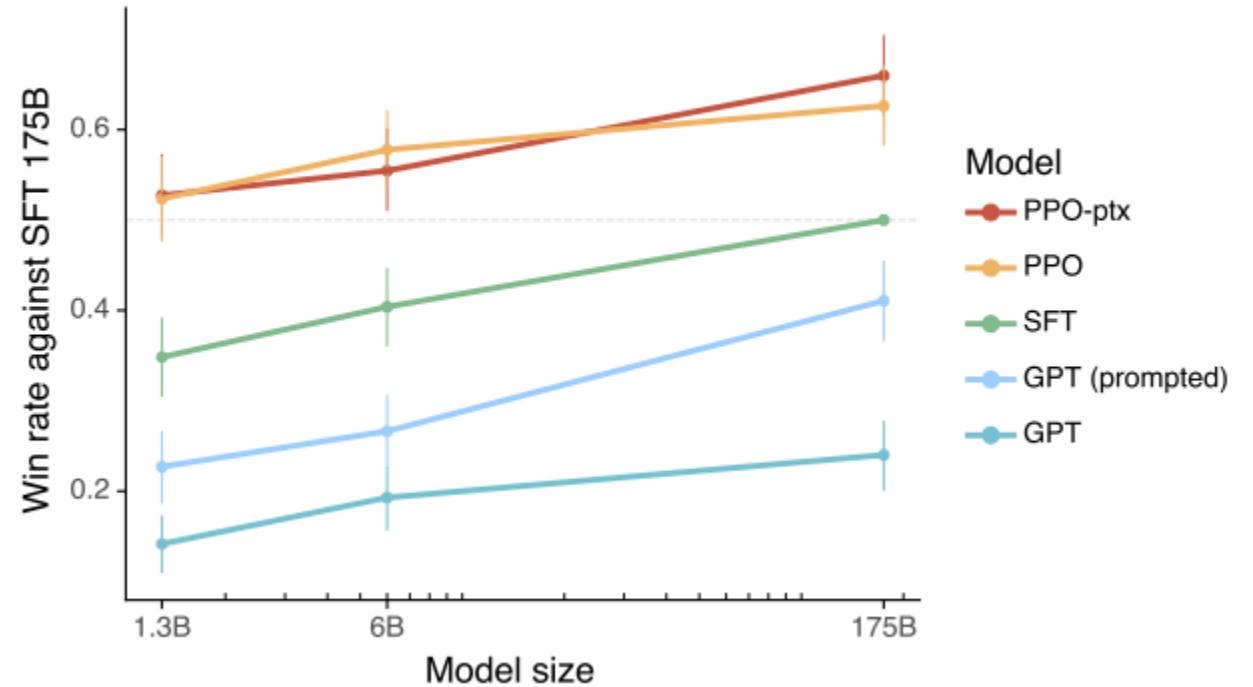
*"As the science of training large language models progresses, certain abilities may be unlocked for smaller models with new architectures, higher-quality data, or improved training procedures"*

- Another potentially way of unlocking emergence is through a different pre-training objective



# Emergence: Finetuning for Desired Behaviors

- Desired behaviors can be induced in smaller models via finetuning and RLHF



# Emergent Risks

- Social risks -
  - Truthfulness
  - Bias
  - Toxicity
- Other risks -
  - Backdoor vulnerabilities
  - Inadvertent deception
  - Harmful content synthesis

# Future Directions

- Further model scaling
- Improved model architectures and training
- Data scaling
- Better techniques for and understanding of prompting
- Frontier tasks
- Understanding emergence

# Summary & Conclusion

- Emergent abilities can span a variety of language models, task types, and experimental scenarios
- Model scale is not the singular factor for unlocking an emergent ability
- The questions on how LLMs emerge and whether more scaling will enable further emergent abilities seem to be important future research directions for the field of NLP

# References

- <https://stackoverflow.com/questions/64485777/how-is-the-number-of-parameters-be-calculated-in-bert-model>
- <https://github.com/sovrasov/flops-counter.pytorch>
- <https://ai.googleblog.com/2022/11/characterizing-emergent-phenomena-in.html>
- [https://www.stat.cmu.edu/~ryantibs/convexopt-F18/scribes/Lecture\\_19.pdf](https://www.stat.cmu.edu/~ryantibs/convexopt-F18/scribes/Lecture_19.pdf)
- <https://stackoverflow.com/questions/58498651/what-is-flops-in-field-of-deep-learning>
- <https://stackoverflow.com/questions/55831235/calculating-the-number-of-flops-for-a-given-neural-network>
- <https://openreview.net/pdf?id=l1w0Gj8v6Kd>