Deep Learning

Lecture 8: GPT and Large Language Models

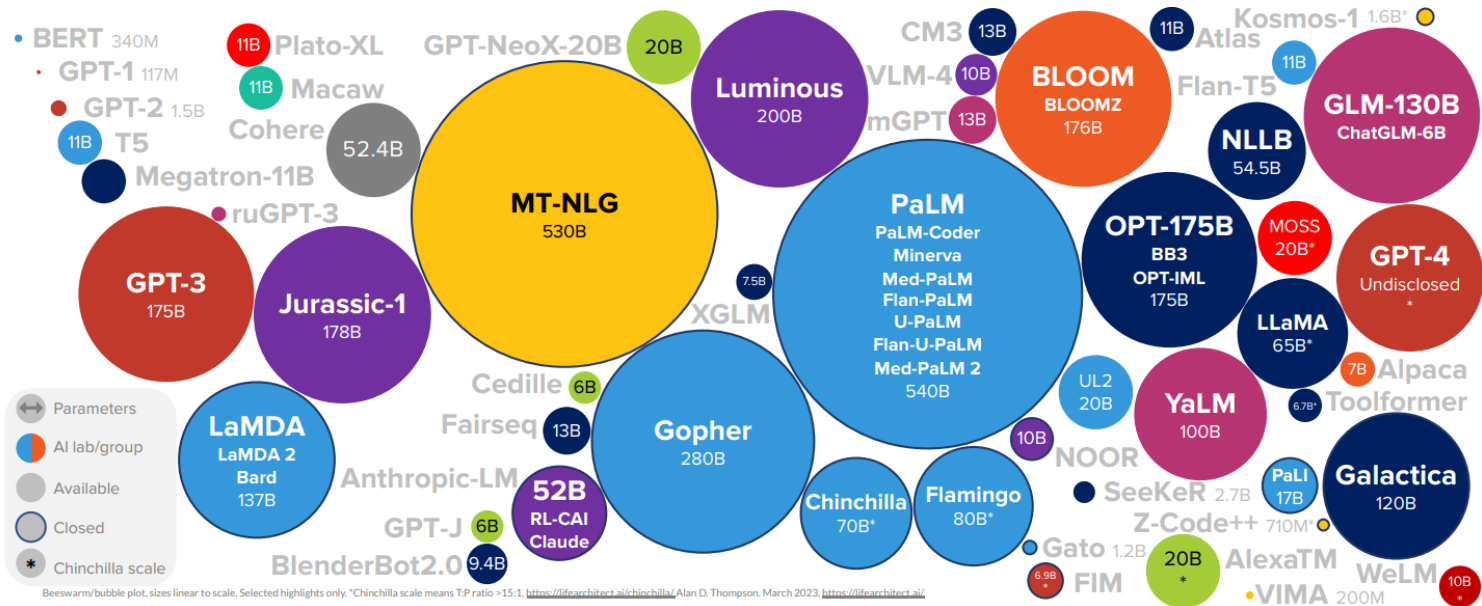Prof. Gilles Louppe

g.louppe@uliege.be

LIÈGE université

Today

- BabyGPT

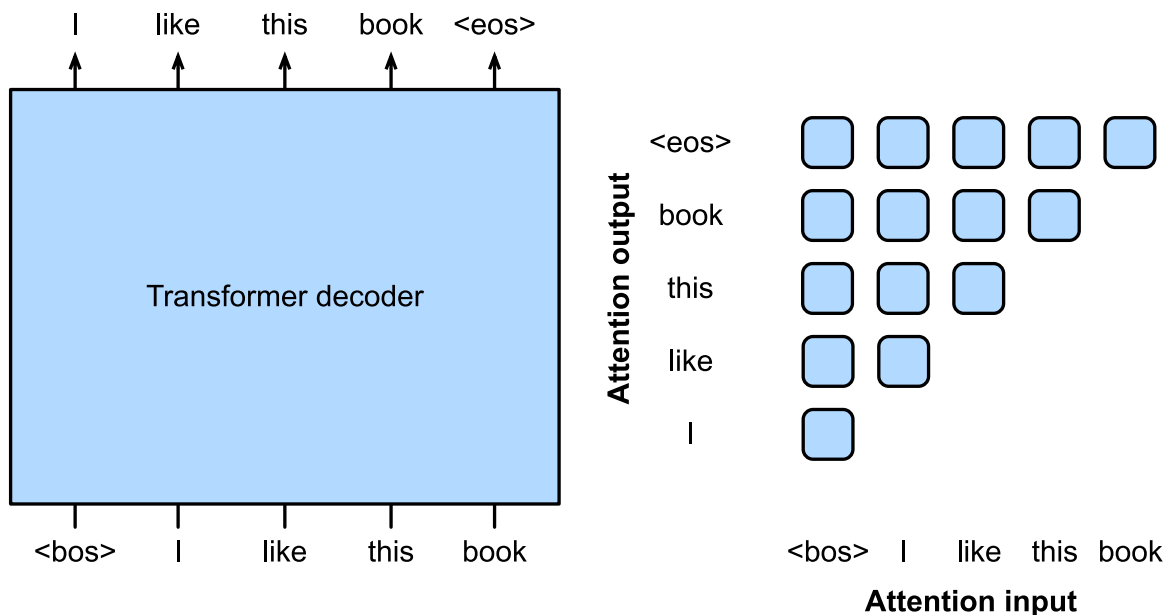- Large language models

See `code/gpt/`.

Large language models

(March 2023)

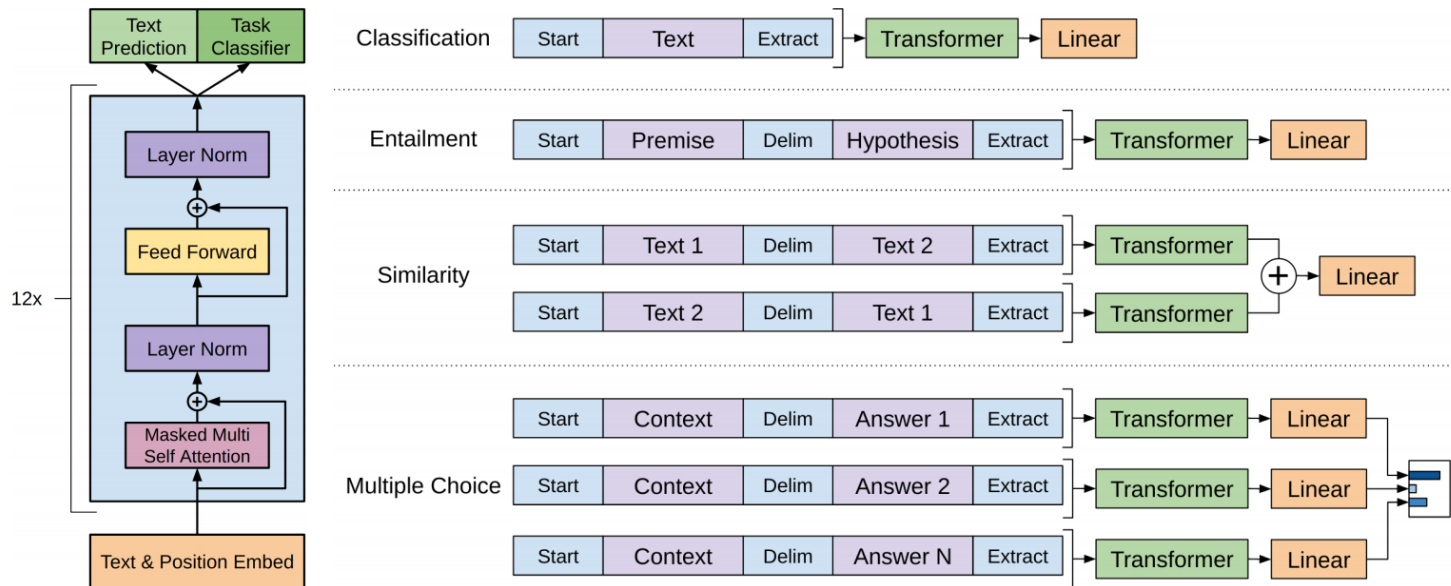# Decoder-only transformers

The decoder-only transformer has become the de facto architecture for large language models.

These models are trained with self-supervised learning, where the target sequence is the same as the input sequence, but shifted by one token to the right.

Historically, GPT-1 was first pre-trained and then fine-tuned on downstream tasks.

Scaling laws

Transformer language model performance improves smoothly as we increase the model size, the dataset size, and amount of compute used for training.
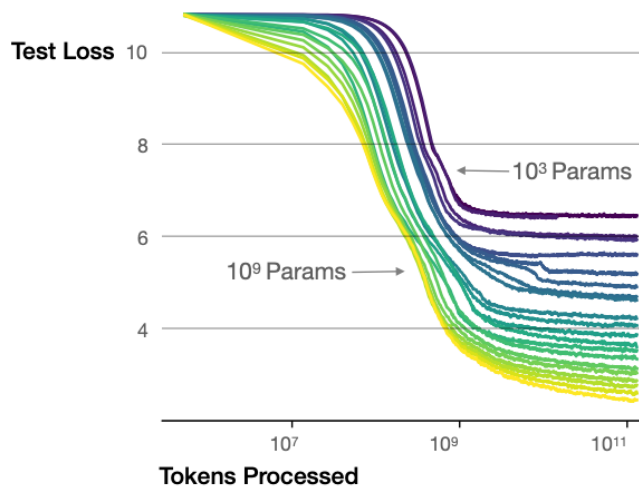
For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.
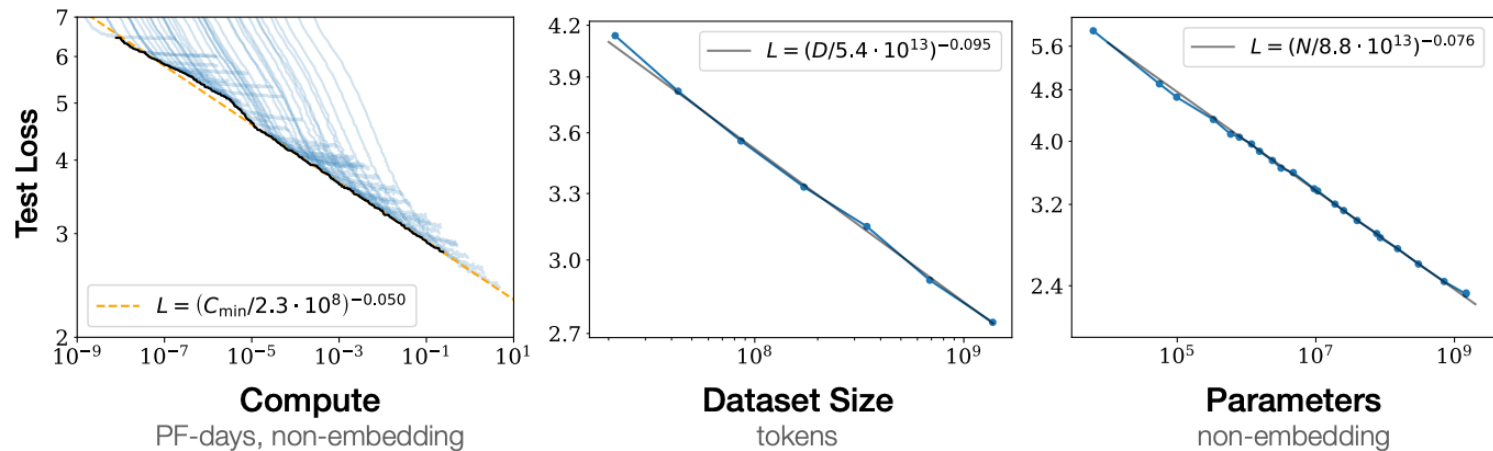
Large models also enjoy better sample efficiency than small models. Larger models require less data to achieve the same performance. The optimal model size also shows to grow smoothly with the amount of compute available for training.

Larger models require **fewer samples** to reach the same performance

The optimal model size grows smoothly with the loss target and compute budget

Line color indicates number of parameters

10^3 10^6 10^9

Test Loss

10^3 Params

10^9 Params

Compute-efficient training stops far short of convergence

Tokens Processed

Compute (PF-days)
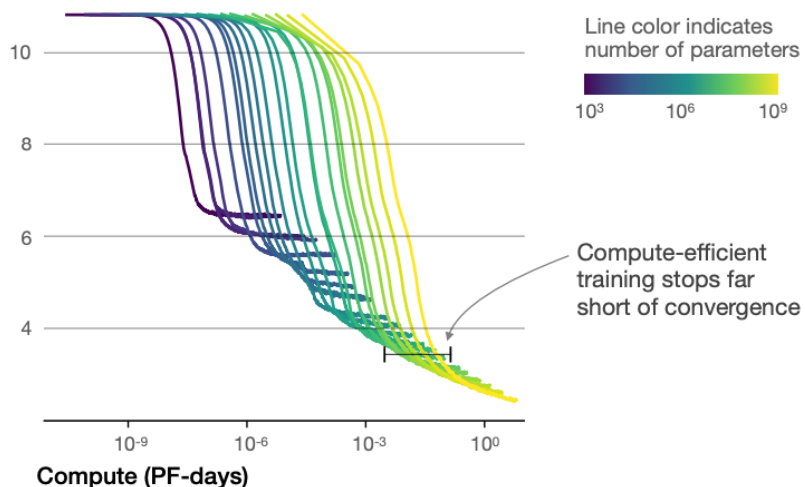
In-context learning

GPT-2 and following models demonstrated potential of using the same language model for multiple tasks, **without updating the model weights**.

Zero-shot, one-shot and few-shot learning consist in prompting the model with a few examples of the target task and letting it learn from them. This paradigm is called in-context learning.

Zero-shot

Output   je suis malade

Transformer decoder
(no parameter update)

Input   Translate English to French: i'm home ->

← – – – – – – – – – – → ← – – – →
Task description   *Prompt*

One-shot

Output   je suis malade

Transformer decoder
(no parameter update)

Input   Translate English to French: **go -> va |** i'm home ->

← – – – – – – – – – – → ← – – → ← – – →
Task description   ***One example***   *Prompt*

Few-shot

Output   je suis malade

Transformer decoder
(no parameter update)

Input   Translate English to French: **go -> va | i lost -> j'ai perdu | he's calm -> elle court |** i'm home ->

← – – – – – – – – – – → ← – – – – – – – – – – – – – – – – – → ← – – – →
Task description   ***A few examples***   *Prompt*
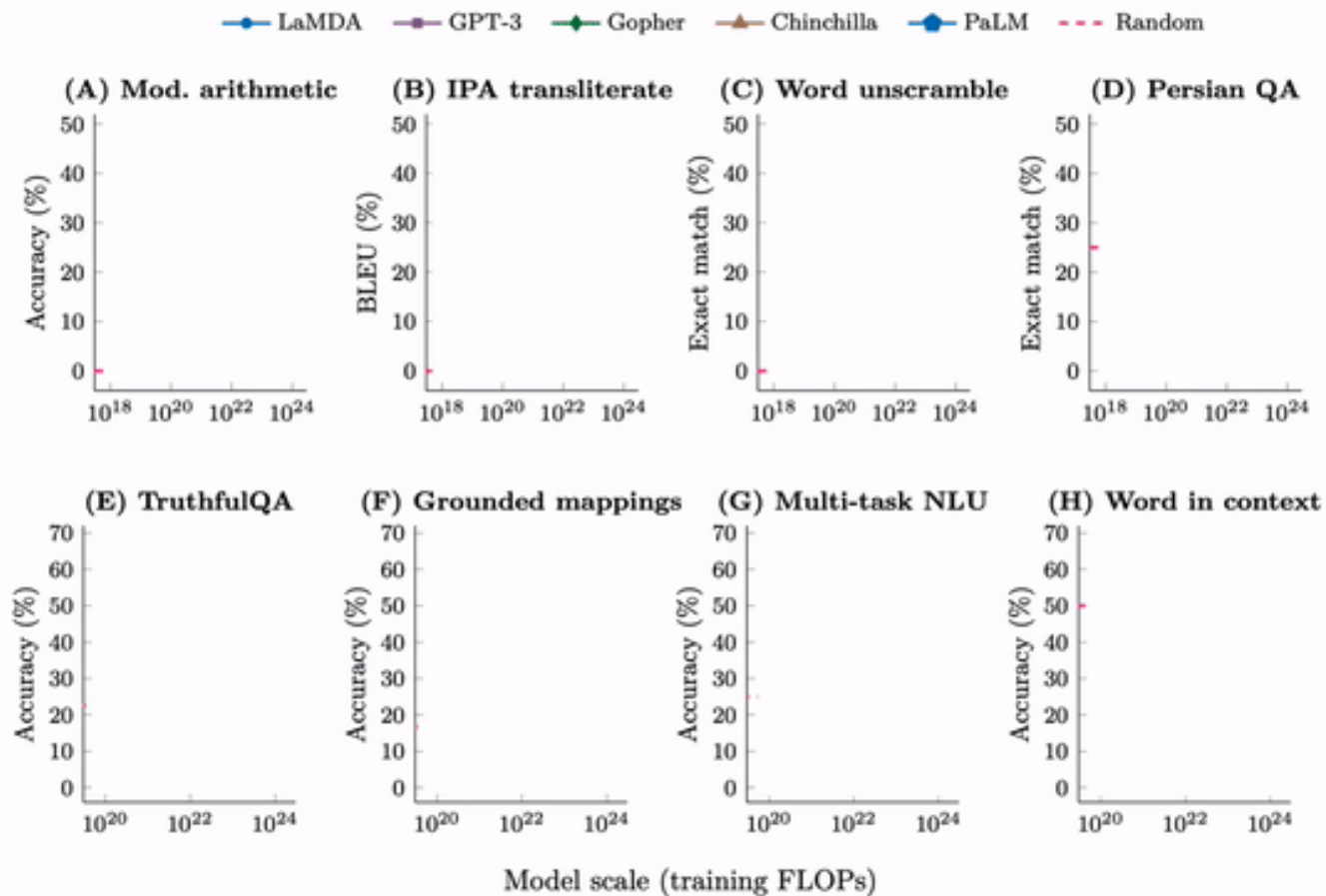
Demo of Llama 7B vs GPT-3.5 vs GPT-4

Emergent abilities

As language models grow in size, they start to exhibit emergent abilities that are not present in the original training data.

A (few-shot) prompted task is **emergent** if it achieves random performance for small models and then (suddenly) improves as the model size increases.

Legend: LaMDA · GPT-3 · Gopher · Chinchilla · PaLM · Random

(A) Mod. arithmetic — Accuracy (%)
(B) IPA transliterate — BLEU (%)
(C) Word unscramble — Exact match (%)
(D) Persian QA — Exact match (%)
(E) TruthfulQA — Accuracy (%)
(F) Grounded mappings — Accuracy (%)
(G) Multi-task NLU — Accuracy (%)
(H) Word in context — Accuracy (%)

Model scale (training FLOPs)

Notably, chain-of-thought reasoning is an emergent ability of large language models. It improves performance on a wide range of arithmetica, commonsense, and symbolic reasoning tasks.

Standard Prompting

Model 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

Model 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. ✔

Standard prompting
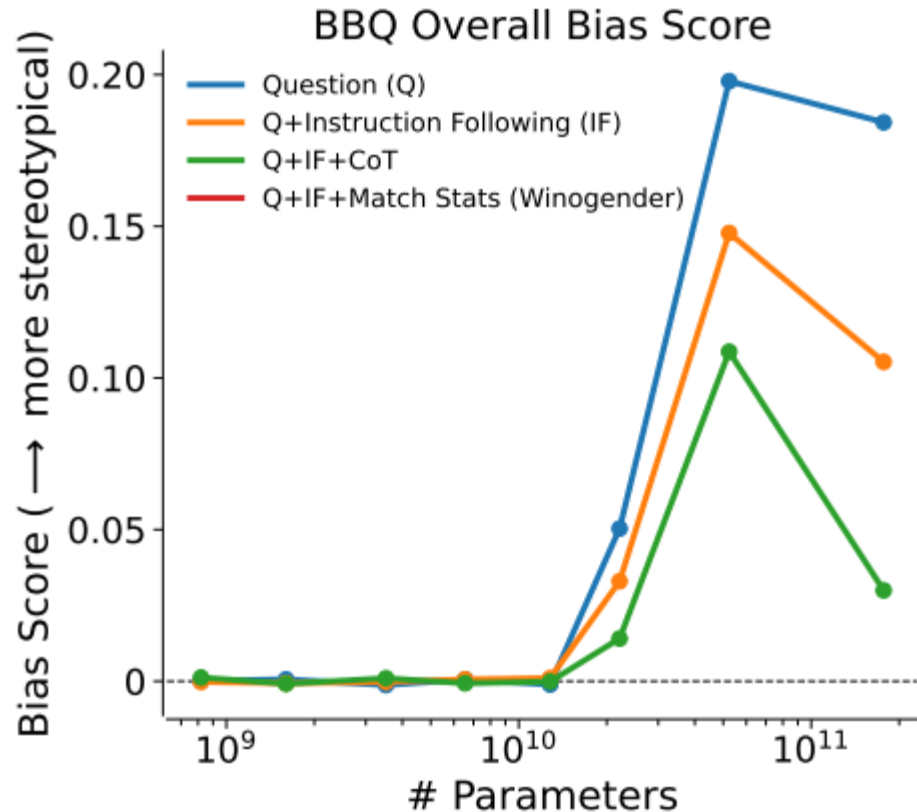Chain-of-thought prompting
Prior supervised best
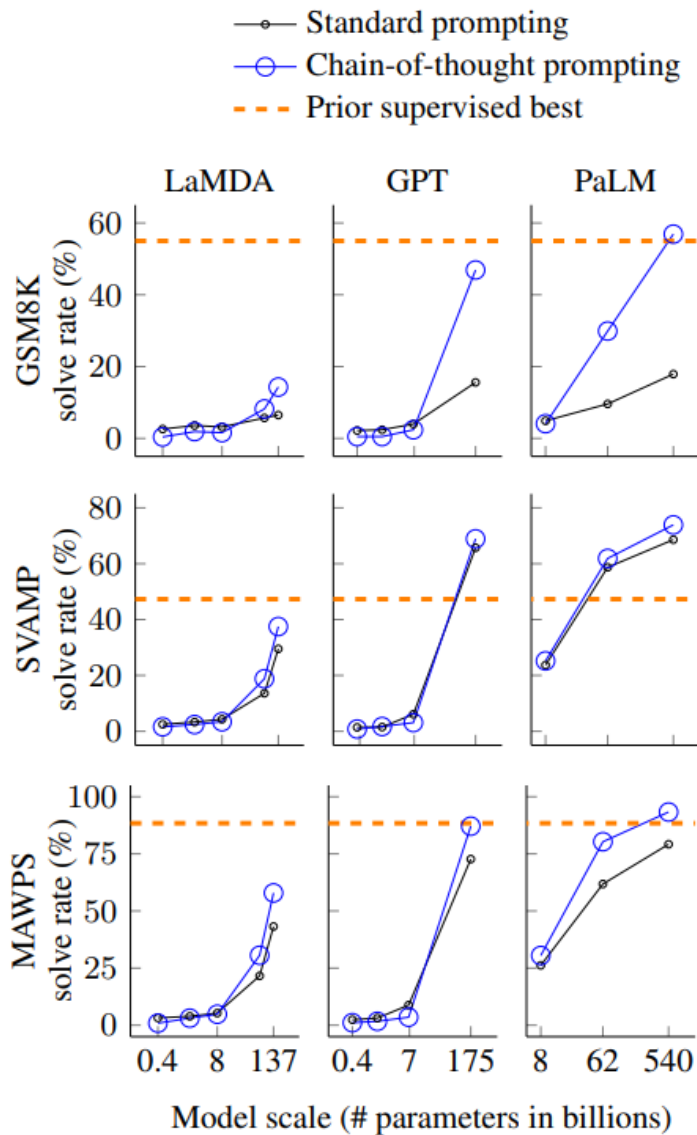
Alignment

Increasing the model size does not inherently makes models follow a user's intent better, despite emerging abilities.

Worse, scaling up the model may increase the likelihood of undesirable behaviors, including those that are harmful, unethical, or biased.

BBQ Overall Bias Score

Legend:
- Question (Q)
- Q+Instruction Following (IF)
- Q+IF+CoT
- Q+IF+Match Stats (Winogender)

Y-axis: Bias Score (⟶ more stereotypical)

X-axis: # Parameters

Human feedback can be used for aligning language models with human intent, as shown by InstructGPT.
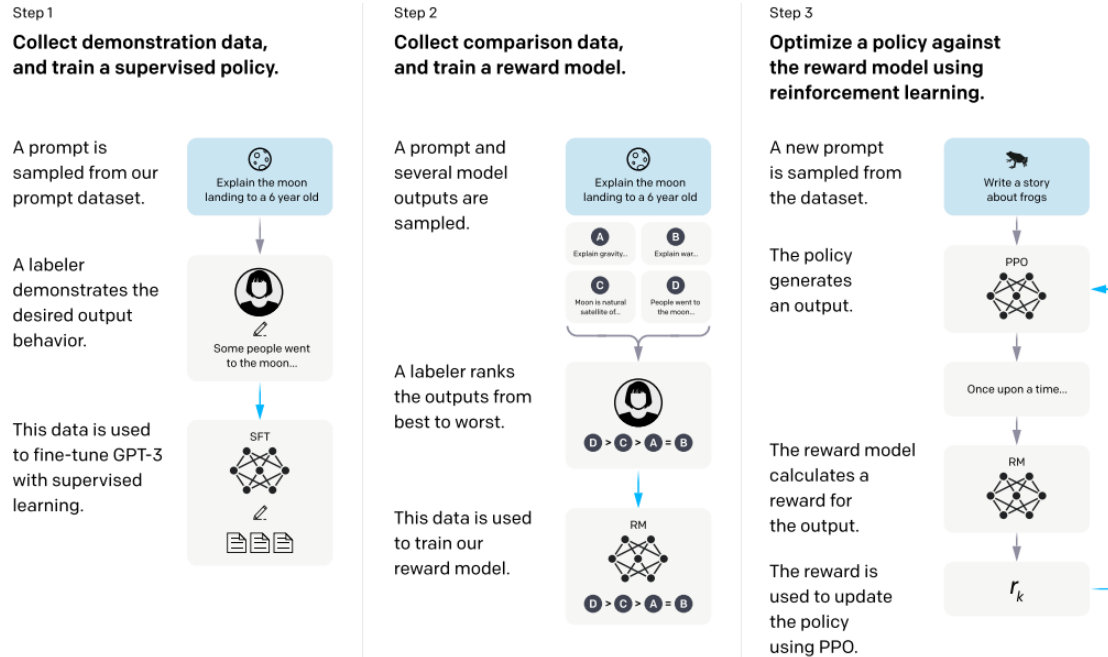

Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

Prompt:

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

GPT-3 175B completion:

A. to store the value of C[0]

B. to store the value of C[1]

C. to store the value of C[i]

D. to store the value of C[i - 1]

InstructGPT 175B completion:

The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

The end.