



LLM Scaling Laws & Reasoning

Yu Meng

University of Virginia
yumeng5@virginia.edu

Oct 21, 2024

Announcement

Join at
slido.com
#4260 208



- Assignment 3 grades posted; reference answer released
- Contact Wenqian (pvc7hs@virginia.edu) if you have questions about your grade



Overview of Course Contents

- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling and Neural Language Models
- Week 6-7: Language Modeling with Transformers (Pretraining + Fine-tuning)
- Week 8: Large Language Models (LLMs) & In-context Learning
- **Week 9-10: Reasoning, Knowledge, and Retrieval-Augmented Generation (RAG)**
- Week 11: LLM Alignment
- Week 12: Language Agents
- Week 13: Recap + Future of NLP
- Week 15 (after Thanksgiving): Project Presentations



(Recap) Large Language Models (LLMs)

- The field of LLMs is rapidly evolving!
 - In 2018, BERT-large with 340 million parameters was considered large
 - In 2019, GPT-2 with 1.5 billion parameters was considered very large
 - In 2020, GPT-3 with 175 billion parameters set a new standard for “large”
- In 2024, how should we define LLMs?
- General definition:
 - Transformer-decoder architecture (or variants) that can generate text
 - Pretrained on vast and diverse general-domain corpora
 - With (at least) billions of parameters
 - General-purpose solvers for a wide range of NLP tasks and beyond



(Recap) Prompting

- **Prompt:** initial user input/instructions given to the model to guide text generation
- Example (sentiment analysis):

$P(\text{positive} | \text{The sentiment of the sentence "I like Jackie Chan" is :})$
 $P(\text{negative} | \text{The sentiment of the sentence "I like Jackie Chan" is :})$

prompt

- Example (question answering):

$P(w | \text{Q: Who wrote the book "The Origin of Species"? A:})$

prompt

- **Prompting:** directly use trained LMs to generate text given user prompts (no fine-tuning)

For good prompting performance, we need **instruction-tuning** (later lectures)



(Recap) Parameter Efficient Fine-tuning (PEFT)

- Fine-tuning all model parameters is expensive

Pretrained weight
(can represent any module) $\mathbf{W}_0 \in \mathbb{R}^{d \times d}$

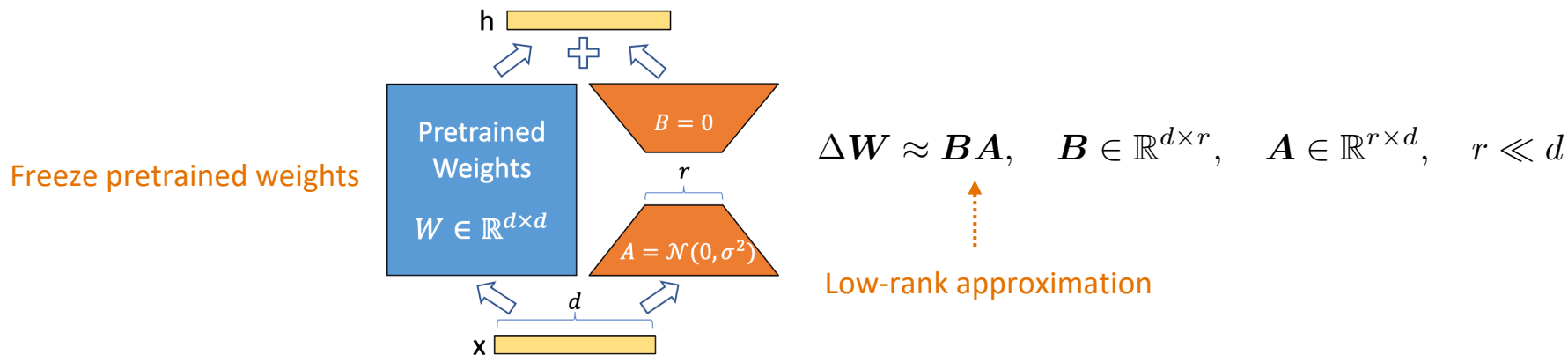
Fine-tuned weight $\mathbf{W}^* = \mathbf{W}_0 + \Delta\mathbf{W}$, $\Delta\mathbf{W} \in \mathbb{R}^{d \times d}$

- Can we update only a small number of model parameters on fine-tuning data?



(Recap) Parameter Efficient Fine-tuning: LoRA

- Assume the parameter update is **low-rank**
 - Overparameterization**: large language models typically have many more parameters than strictly necessary to fit the training data
 - Empirical observation**: parameter updates in neural networks tend to be low-rank in practice
- Solution: approximate weight updates with low-rank factorization





(Recap) Decoding with LLMs

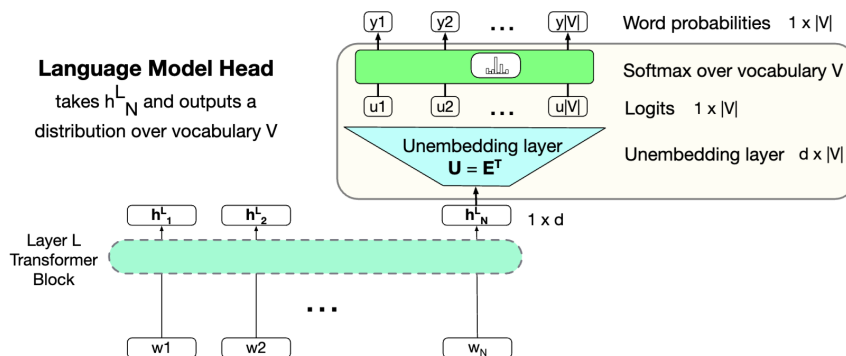
- **Decoding:** convert Transformer representations into natural language tokens
- Autoregressive decoding typically involves iterative **sampling** from LMs' output distributions, until an [EOS] token is generated

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}\mathbf{h}_{i-1}) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1})}, \dots, \frac{\exp(\mathbf{u}_{|\mathcal{V}|} \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1})} \right]$$

Model parameters

Unembedding matrix

Hidden states at token $i - 1$





(Recap) Greedy Decoding

- Always pick the token with the highest probability estimated by the LM for every step

$$x_i \leftarrow \arg \max_w p_{\theta}(w | x_1, x_2, \dots, x_{i-1})$$

- Pros:
 - Simplicity: easy to implement and understand
 - Deterministic: guarantee the same output given the same input
 - Efficient: make only one (simple) decision at each step w/o additional operations
- Cons:
 - Suboptimal solutions: may not find the globally optimal sequence
 - Lack of diversity: cannot produce multiple outputs given the same input



(Recap) Top- k Sampling

- Motivation: Instead of choosing the single most probable word to generate, sample from the top- k most likely tokens (candidates) – avoid generating low probability tokens
- k is a hyperparameter (typically 5-10)

Compute the probability distribution only over the top- k tokens

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}_{\text{top-}k} \mathbf{h}_{i-1}) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^k \exp(\mathbf{u}_{\text{top-}j} \cdot \mathbf{h}_{i-1})}, \dots, \frac{\exp(\mathbf{u}_{\text{top-}k} \cdot \mathbf{h}_{i-1})}{\sum_{j=1}^k \exp(\mathbf{u}_{\text{top-}j} \cdot \mathbf{h}_{i-1})} \right]$$

Sample from the top- k tokens $x_i \sim p_{\theta}(w|x_1, x_2, \dots, x_{i-1})$

- With $k = 1$, top- k sampling is equivalent to greedy decoding



(Recap) Nucleus (Top- p) sampling

- Top- k sampling does not account for the shape of the probability distribution
 - For the next-token distribution of “the 46th US president Joe”, top- k sampling may consider more tokens than necessary
 - For the next-token distribution of “the spacecraft”, top- k sampling may consider fewer tokens than necessary
- Nucleus sampling sets cutoff based on the top- p percent of the probability mass
- p is a hyperparameter (typically 0.9)
- Top- p vocabulary is the smallest set of words such that

$$\sum_{w \in \mathcal{V}_{\text{top-}p}} p(w|x_1, x_2, \dots, x_{i-1}) \geq p$$

- Sample from the top- p vocabulary in a similar way as top- k sampling

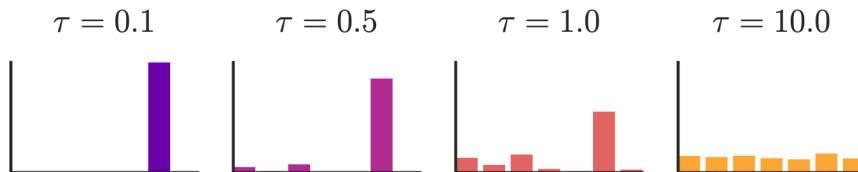


(Recap) Temperature Sampling

- Intuition comes from thermodynamics
 - A system at a high temperature is flexible and can explore many possible states
 - A system at a lower temperature is likely to explore a subset of lower energy (better) states
- Reshape the probability distribution by incorporating a temperature hyperparameter

$$p_{\theta}(w|x_1, x_2, \dots, x_{i-1}) = \text{softmax}(\mathbf{U}\mathbf{h}_{i-1}/\tau) = \left[\frac{\exp(\mathbf{u}_1 \cdot \mathbf{h}_{i-1}/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1}/\tau)}, \dots, \frac{\exp(\mathbf{u}_{|\mathcal{V}|} \cdot \mathbf{h}_{i-1}/\tau)}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathbf{u}_j \cdot \mathbf{h}_{i-1}/\tau)} \right]$$

- With $\tau \rightarrow 0$, temperature sampling approaches greedy decoding





(Recap) Practical Considerations of Decoding

- If aiming for simplicity and efficiency without diversity requirements, use greedy decoding
- If multiple responses are required for the same input, use sampling-based decoding
 - Top- p is usually better than Top- k
 - Temperature sampling is commonly used
 - Top- p can be used together with temperature sampling

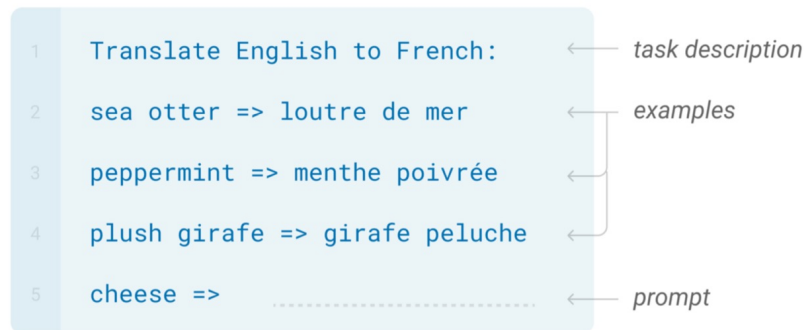


(Recap) In-context Learning

- In-context learning is a type of few-shot learning
 - User provides a few examples of input-output pairs in the prompt
 - The model uses given examples to predict the output for new, similar inputs
- First studied in the GPT-3 paper
- No model parameter updates

Few-shot

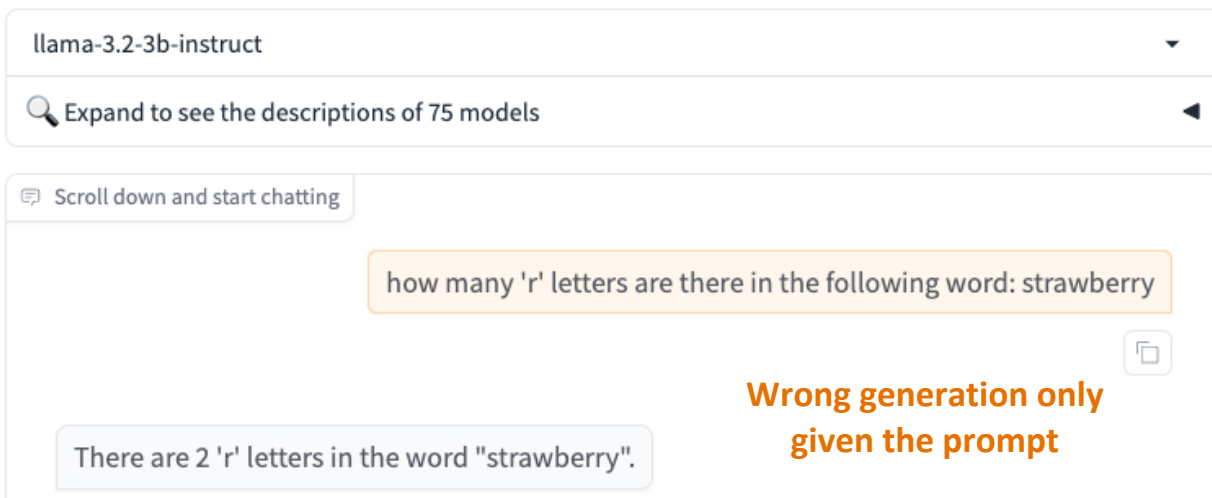
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.





(Recap) In-context Learning Demo

Prompt: how many 'r' letters are there in the following word: strawberry



llama-3.2-3b-instruct

Expand to see the descriptions of 75 models

Scroll down and start chatting

how many 'r' letters are there in the following word: strawberry

There are 2 'r' letters in the word "strawberry".

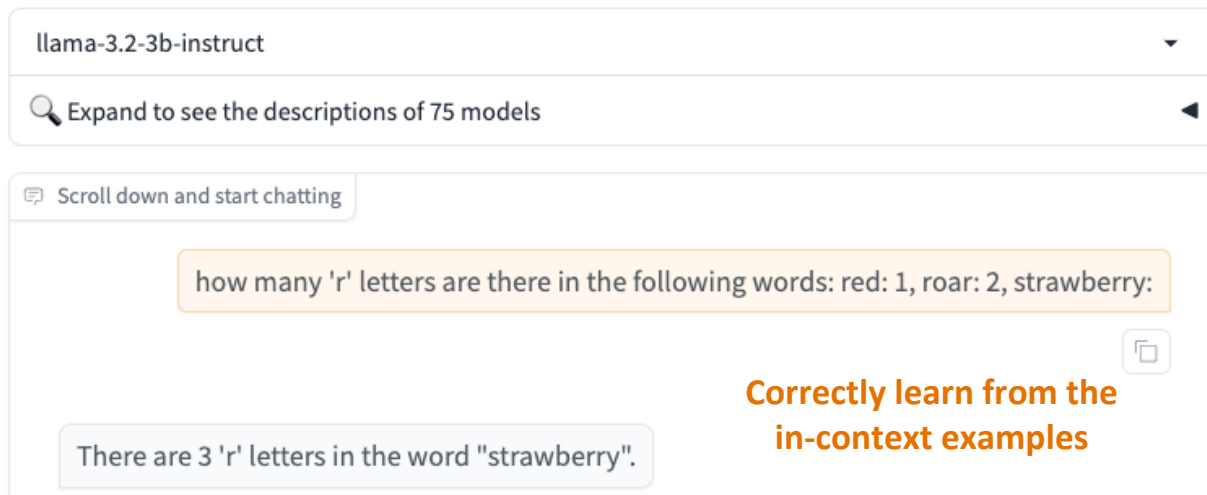
Wrong generation only given the prompt

Generated with greedy decoding
(temperature = 0)



(Recap) In-context Learning Demo

Prompt: how many 'r' letters are there in the following words: red: 1, roar: 2, strawberry:



The screenshot shows a chat interface with a model name 'llama-3.2-3b-instruct' at the top. Below it is a search bar with the text 'Expand to see the descriptions of 75 models'. A button labeled 'Scroll down and start chatting' is visible. The main chat area contains a user prompt: 'how many 'r' letters are there in the following words: red: 1, roar: 2, strawberry:'. Below the prompt is a copy icon. The model's response is: 'There are 3 'r' letters in the word "strawberry".'

**Correctly learn from the
in-context examples**

Generated with greedy decoding
(temperature = 0)

Agenda

- Scaling Up LLMs
- Chain-of-thought Reasoning

Join at
slido.com
#4260 208





Scaling Up Pretraining Data

The Pile: 22 sub-datasets (> 800GB), a common choice for pretraining corpus
 Composition of the Pile by Category

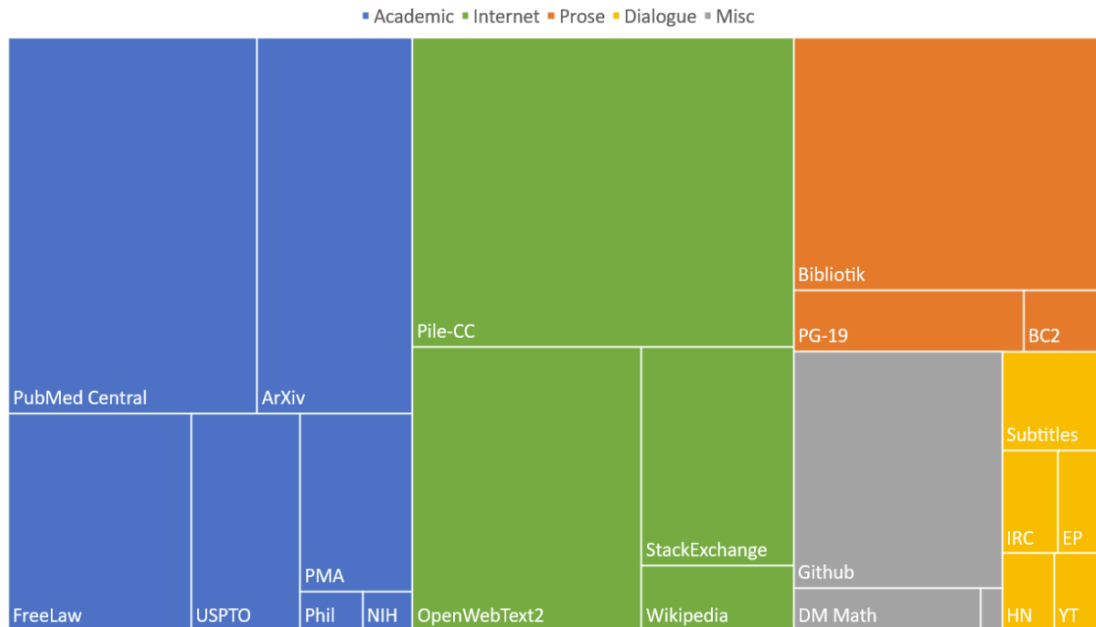


Figure source: <https://arxiv.org/pdf/2101.00027>



Broader Range of Knowledge by Scaling Up Data

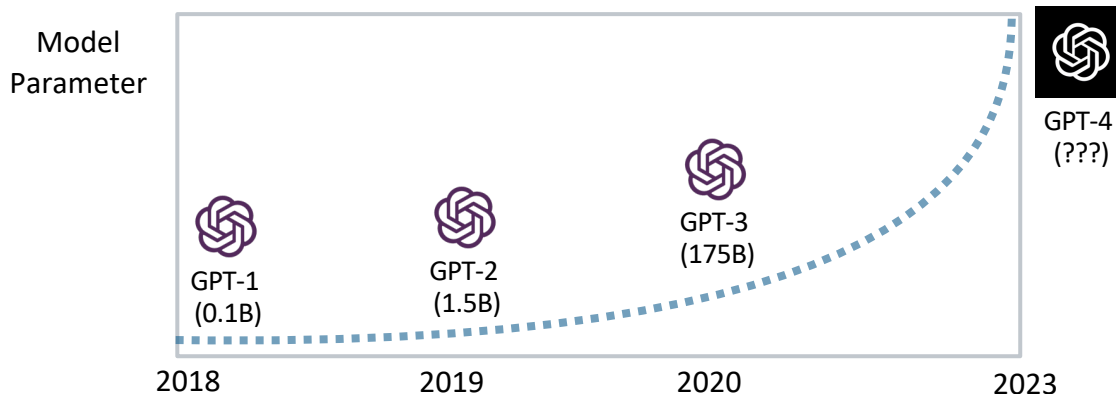
- In my free time, I like to **{run, banana}** (*Grammar*)
- I went to the zoo to see giraffes, lions, and **{zebras, spoon}** (*Lexical semantics*)
- The capital of Denmark is **{Copenhagen, London}** (*World knowledge*)
- I was engaged and on the edge of my seat the whole time. The movie was **{good, bad}** (*Sentiment analysis*)
- The word for “pretty” in Spanish is **{bonita, hola}** (*Translation*)
- $3 + 8 + 4 = \mathbf{\{15, 11\}}$ (*Math*)
- ...

Examples from: https://docs.google.com/presentation/d/1hQUd3pF8_2Gr2Obc89LKjmHLODIH-uof9M0yFVd3FA4/edit#slide=id.g28e2e9aa709_0_1



Scaling Up Model Sizes

- GPT-1 (2018): 12 layers, 117M parameters, trained in ~1 week
- GPT-2 (2019): 48 layers, 1.5B parameters, trained in ~1 month
- GPT-3 (2020): 96 layers, 175B parameters, trained in several months



Papers: (GPT-1) https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

(GPT-2) https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

(GPT-3) <https://arxiv.org/pdf/2005.14165.pdf>

Emergent Ability

Join at
slido.com
#4260 208



- Larger models develop **emergent abilities**
 - Skills or capabilities that were not explicitly learned but arise as a result of model capacity
 - Larger models demonstrate surprising abilities in challenging tasks even when they were not explicitly trained for them
- Emergent capabilities typically become noticeable only when the model size reaches a certain threshold (cannot be predicted by small model's performance)

Emergent Abilities of Large Language Models

Jason Wei¹
Yi Tay¹
Rishi Bommasani²
Colin Raffel³
Barret Zoph¹
Sebastian Borgeaud⁴
Dani Yogatama⁴
Maarten Bosma¹
Denny Zhou¹
Donald Metzler¹
Ed H. Chi¹
Tatsunori Hashimoto²
Oriol Vinyals⁴
Percy Liang²
Jeff Dean¹
William Fedus¹

jasonwei@google.com
ytay@google.com
nlprishi@stanford.edu
craffel@gmail.com
barretzoph@google.com
sborgeaud@deepmind.com
dyogatama@deepmind.com
bosma@google.com
dennyzhou@google.com
metzler@google.com
edchi@google.com
thashim@stanford.edu
vinyals@deepmind.com
pliang@stanford.edu
jeff@google.com
liamfedus@google.com

Paper: <https://arxiv.org/pdf/2206.07682>

¹Google Research ²Stanford University ³UNC Chapel Hill ⁴DeepMind

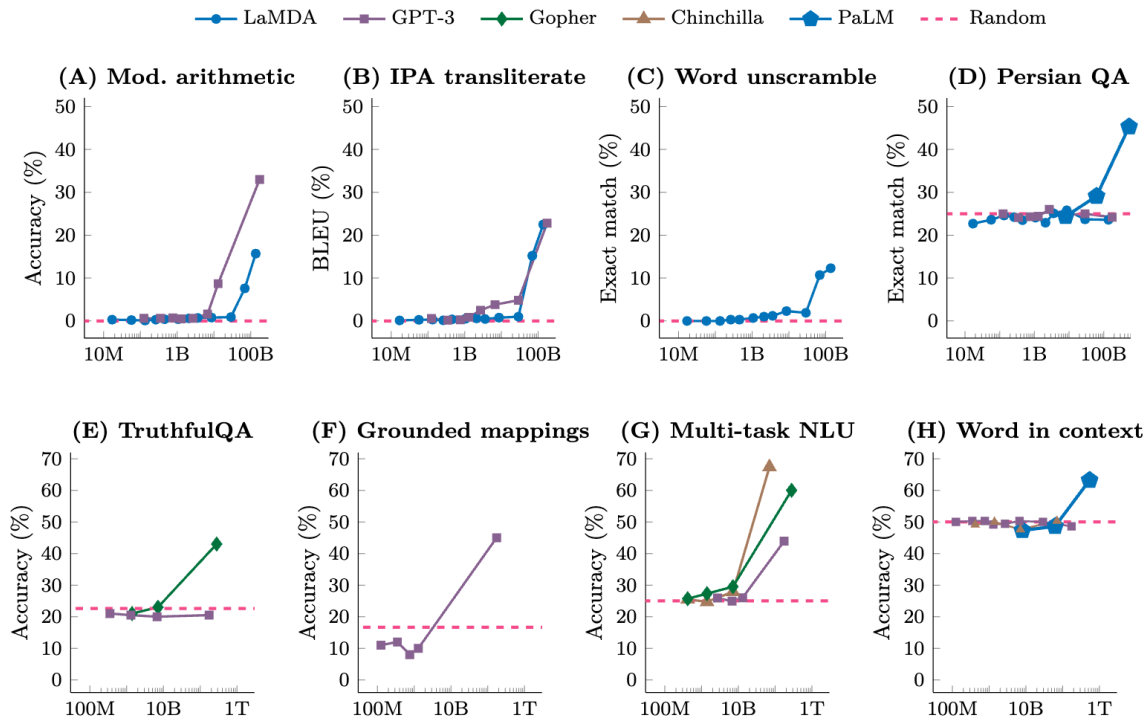
Experiment Setting



- Consider the **few-shot in-context learning** paradigm
- Consider an ability to be **emergent** when a model has **random** performance until a certain scale, after which performance increases to **well-above random**
- Abilities to test
 - Arithmetic: addition, subtraction, multiplication
 - Transliteration
 - Recover a word from its scrambled letters
 - Persian question answering
 - Question answering (truthfully)
 - Grounded conceptual mappings
 - Multi-task understanding (math, history, law, ...)
 - Contextualized semantic understanding



Performance vs. Model Scale



Model scale (number of parameters)

Models exhibit random performance until a certain scale, after which performance significantly increases



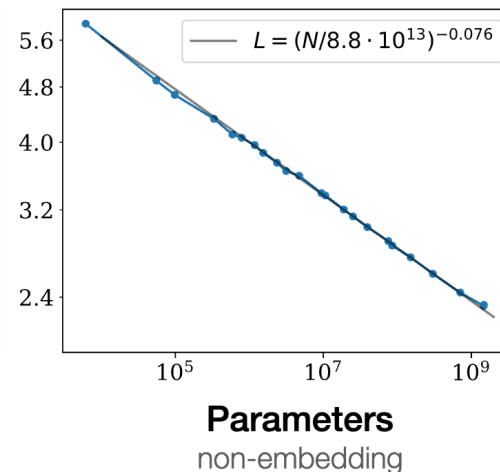
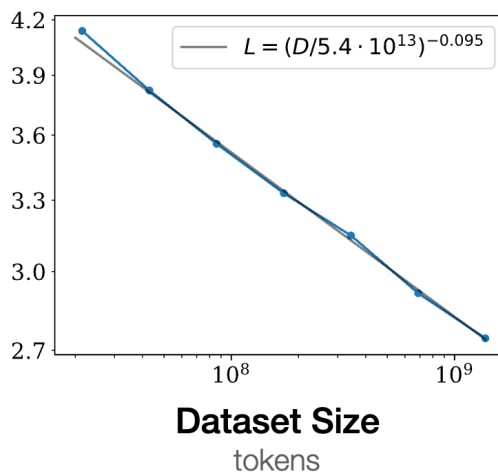
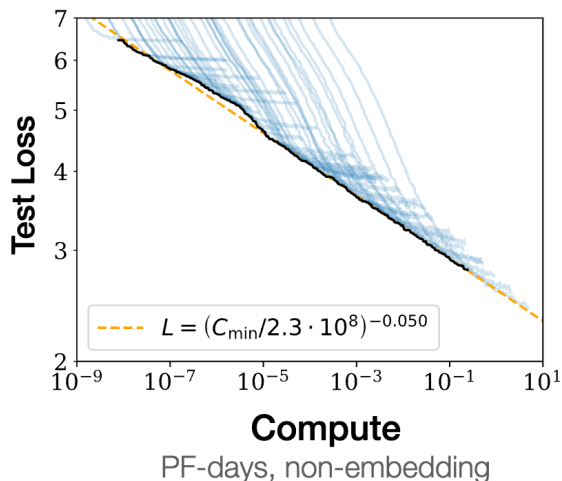
Scaling Laws of LLMs

- (Pretrained) LLM performance is mainly determined by 3 factors
 - Model size: the number of parameters
 - Dataset size: the amount of training data
 - Compute: the amount of floating point operations (FLOPs) used for training
- Scaling up LLMs involves scaling up the 3 factors
 - Add more parameters (adding more layers or having more model dimensions or both)
 - Add more data
 - Train for more iterations
- **Scaling laws:** study the correlation between the cross-entropy language modeling loss and the above three factors
- How to optimally allocate a fixed compute budget?



Scaling Laws of LLMs

Performance has a power-law relationship with each of the three scale factors (model size, dataset size, compute) when not bottlenecked by the other two



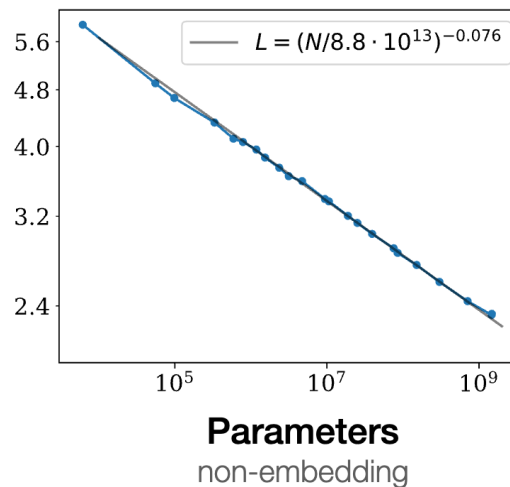


Scaling Model Parameters

- Language model loss vs. models with a limited number of parameters (N)
 - Only count non-embedding parameters
 - Infinite compute: trained to convergence
 - Infinite dataset: trained with sufficiently large datasets
- Performance depends strongly on scale, weakly on model shape (depth vs. width)

$$\mathcal{L}(N) = \left(\frac{N_c}{N} \right)^{\alpha_N}, \quad \alpha_N \approx 0.076, \quad N_c \approx 8.8 \times 10^{13}$$


 Model parameters
 (non-embedding)



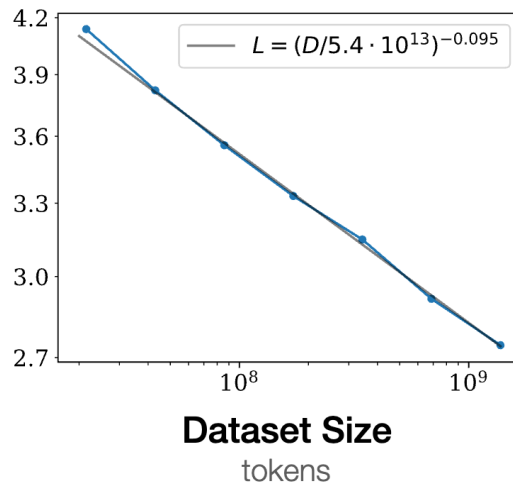


Scaling Dataset Size

- Language model loss vs. a limited dataset size (D)
 - Infinite model size: sufficiently large model
 - With appropriate early stopping: avoid overfitting to the training data

$$\mathcal{L}(D) = \left(\frac{D_c}{D} \right)^{\alpha_D}, \quad \alpha_D \approx 0.095, \quad D_c \approx 5.4 \times 10^{13}$$


 Dataset size
 (# of tokens)



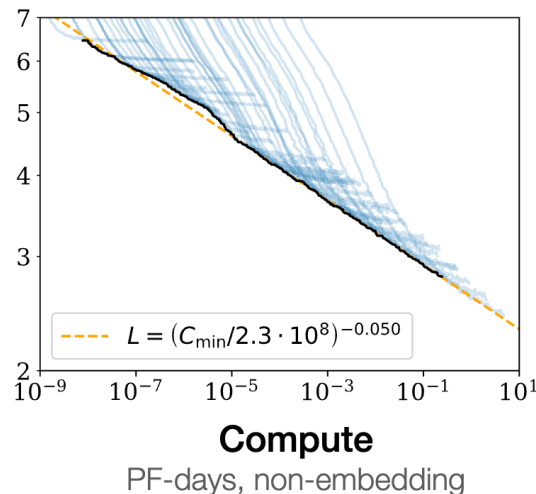


Scaling Training Compute

- Language model loss vs. a limited amount of compute (C)
 - Infinite dataset size: sufficiently large training corpus
 - Optimal model size: can effectively learn the data and not excessively compute-consuming

$$\mathcal{L}(C) = \left(\frac{C_c}{C} \right)^{\alpha_C}, \quad \alpha_C \approx 0.050, \quad C_c \approx 3.1 \times 10^8$$

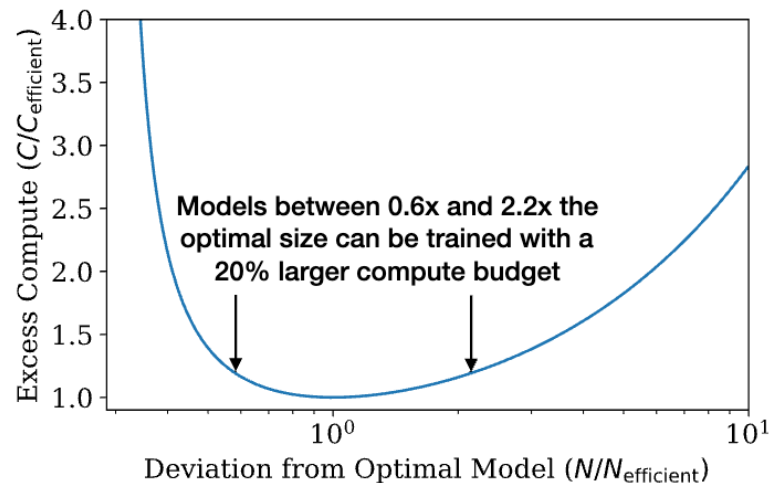
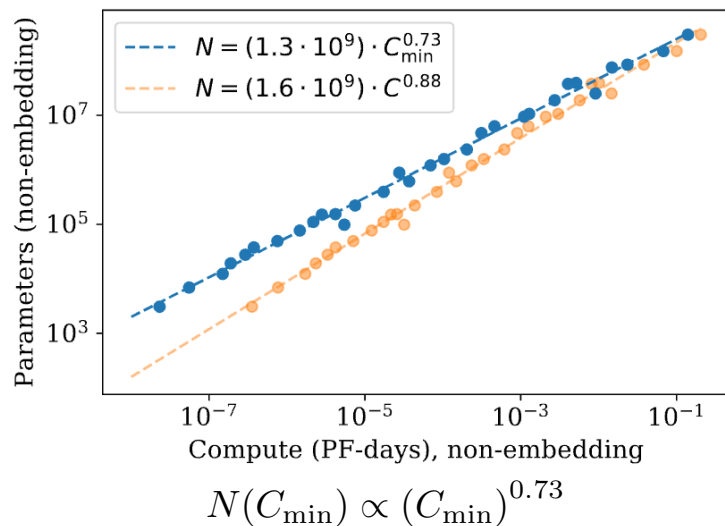

 Compute
 (# Peta-FLOP days)





Optimal Model Size

- Given a specific amount of training compute C , what's the optimal model size $N(C)$ that leads to minimal language modeling loss?
- $N(C)$ can be fit with a power-law wrt C
- Additional compute needs to be used when model size is suboptimal





Further Reading on Scaling LLMs

- [Training Compute-Optimal Large Language Models](#) [Hoffmann et al., 2022]
- [Scaling Data-Constrained Language Models](#) [Muennighoff et al., 2023]
- [Are Emergent Abilities of Large Language Models a Mirage?](#) [Schaeffer et al., 2023]

Agenda

- Scaling Up LLMs
- Chain-of-thought Reasoning

Join at
slido.com
#4260 208





Reasoning: Overview

- **Reasoning** (rough definition): perform deductive, inductive, commonsense, or logical reasoning via generating or analyzing text
- Deductive reasoning: draw specific conclusions from general principles or premises
 - E.g.: “All humans are mortal” + “Socrates is a human” => “Socrates is mortal”
- Inductive reasoning: make generalizations based on specific observations
 - E.g.: “The sun has risen in the east every day” => “The sun will rise in the east tomorrow”
- Commonsense reasoning: rely on world knowledge or commonsense understanding to make predictions or answer questions
 - E.g.: “If I drop a ball, what will happen?” => “It will fall”
- Mathematical/logical reasoning: follow specific rules or procedures to arrive at a correct answer
 - E.g.: “If 3 apples cost \$6, how much do 5 apples cost?” => “\$10”

Latest LLMs for Reasoning: OpenAI o1

Join at
slido.com
#4260 208



September 12, 2024

Learning to Reason with LLMs

We are introducing OpenAI o1, a new large language model trained with reinforcement learning to perform complex reasoning. o1 thinks before it answers —it can produce a long internal chain of thought before responding to the user.

Contributions

OpenAI o1 ranks in the 89th percentile on competitive programming questions (Codeforces), places among the top 500 students in the US in a qualifier for the USA Math Olympiad (AIME), and exceeds human PhD-level accuracy on a benchmark of physics, biology, and chemistry problems (GPQA). While the work needed to make this new model as easy to use as current models is still ongoing, we are releasing an early version of this model, OpenAI o1-preview, for immediate use in ChatGPT and to trusted API users.



OpenAI o1: Commonsense Reasoning

o1-preview

Assume laws of physics on Earth. A small strawberry is put into a normal cup and the cup is placed upside down on a table. Someone then takes the cup and puts it inside the microwave. Where is the strawberry now? Explain your reasoning step by step



Tracking the strawberry's path

To determine the current location of the strawberry, let's break down the events step by step:

1. **Initial Placement:** A small strawberry is placed into a normal cup. The strawberry is inside the cup, resting at the bottom.
2. **Cup Inversion:** The cup is then placed upside down on a table. This means the open end of the cup (where the strawberry was initially placed) is now facing down, in contact with the table surface. The strawberry is now sitting on the table, covered by the inverted cup.
3. **Cup Removal:** Someone picks up the cup to put it inside the microwave. When lifting the inverted cup off the table:



Chain-of-thought (CoT) Prompting

- **Chain-of-thought (CoT):** the model breaks down complex problems into a step-by-step reasoning process
- Instead of directly providing an answer to a question or task, the model is prompted to explain its reasoning or thought process in a logical sequence

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

Google Research, Brain Team
{jasonwei, dennyzhou}@google.com



Standard Prompting vs. CoT Prompting

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?

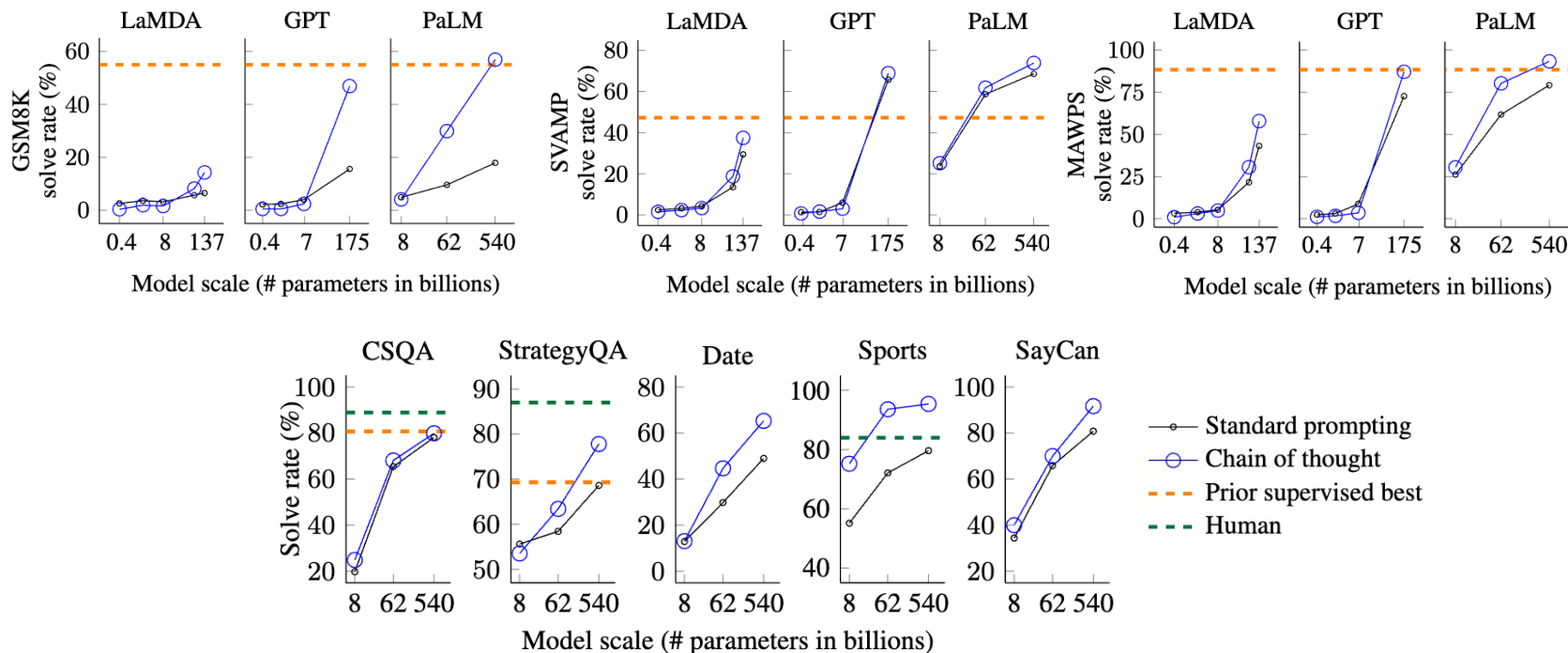
Model Output

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 vs. CoT Prompting Performance

CoT prompting is especially effective for large models





CoT Can Be Triggered Zero-shot

Just add “Let’s think step by step” at the beginning of the answer

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let’s think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

CoT Demo

Join at
slido.com
#4260 208



No-CoT prompt: How many 'r' letters are there in the following word: strawberry? Answer without reasoning steps

llama-3.1-70b-instruct

Expand to see the descriptions of 75 models

Scroll down and start chatting

How many 'r' letters are there in the following word: strawberry? Answer without reasoning steps

There are 2 'r' letters in the word "strawberry". **Wrong result**

CoT prompt: How many 'r' letters are there in the following word: strawberry? Let's think step by step

llama-3.1-70b-instruct

Expand to see the descriptions of 75 models

Scroll down and start chatting

How many 'r' letters are there in the following word: strawberry? Let's think step by step

Step 1: Write down the word
The word is: strawberry

Step 2: Break down the word into individual letters
Here are the individual letters: s-t-r-a-w-b-e-r-r-y

Step 3: Count the number of 'r' letters
Let's count the 'r' letters:

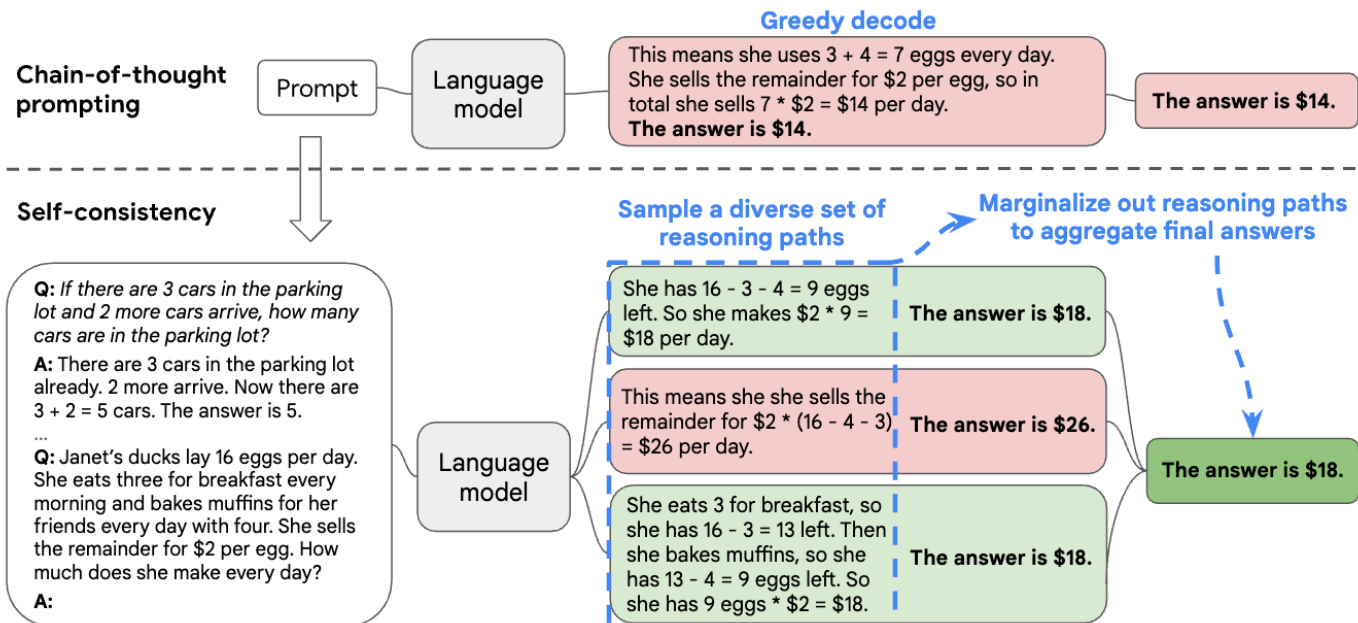
1. r (first 'r')
2. r (second 'r')
3. r (third 'r')

There are 3 'r' letters in the word "strawberry". **Correct result**



Self-consistency CoT

Intuition: if multiple different ways of thinking lead to the same answer, one has greater confidence that the final answer is correct





Self-consistency CoT Demo

- Prompt: When rolling two dice, what is the probability that you roll a total number that is at least 3?

(Previous generation omitted)

- Total of 11: (5,6), (6,5)
- Total of 12: (6,6)

There are 33 favorable outcomes.

Step 3: Calculate the Probability

The probability of rolling a total number that is at least 3 is the number of favorable outcomes divided by the total number of possible outcomes.

$$\begin{aligned}\text{Probability} &= \text{Number of favorable outcomes} / \text{Total number of possible outcomes} \\ &= 33 / 36 \\ &= 11/12\end{aligned}$$

Answer

The probability of rolling a total number that is at least 3 when rolling two dice is 11/12.

Wrong result

(Previous generation omitted)

- Total of 10: (4,6), (5,5), (6,4) = 3 outcomes
- Total of 11: (5,6), (6,5) = 2 outcomes
- Total of 12: (6,6) = 1 outcome

There are 36 possible outcomes, and 35 of them have a total of at least 3.

Step 3: Calculate the Probability

The probability of rolling a total number that is at least 3 is the number of favorable outcomes divided by the total number of possible outcomes.

$$\begin{aligned}\text{Probability} &= \text{Number of favorable outcomes} / \text{Total number of possible outcomes} \\ &= 35 / 36\end{aligned}$$

Answer

The probability of rolling a total number that is at least 3 with two dice is 35/36, which is approximately 0.9722 (or 97.22%).

Correct result

Generated twice with temperature = 0.1, top-p = 0.7
Figure source: <https://lmarena.ai/?model=llama-3.1-70b-instruct>



Further Reading on LLM Reasoning

- [Least-to-Most Prompting Enables Complex Reasoning in Large Language Models](#) [Zhou et al., 2022]
- [Large Language Models Can Self-Improve](#) [Huang et al., 2022]
- [Tree of Thoughts: Deliberate Problem Solving with Large Language Models](#) [Yao et al., 2023]



Thank You!

Yu Meng

University of Virginia

yumeng5@virginia.edu