

Let's train a Tiny Language Model from scratch



Let's train a Tiny Language Model from scratch (PART 1?)



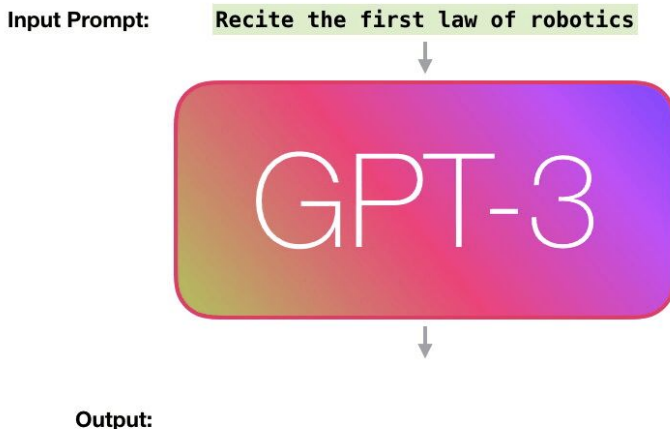
Today's program

How to train a Tiny Language Model from scratch.

1. How are Language Models trained?
2. What are Small and Tiny Language models and why would they be useful?
3. Practical example using **llm.c**

Large Language Models

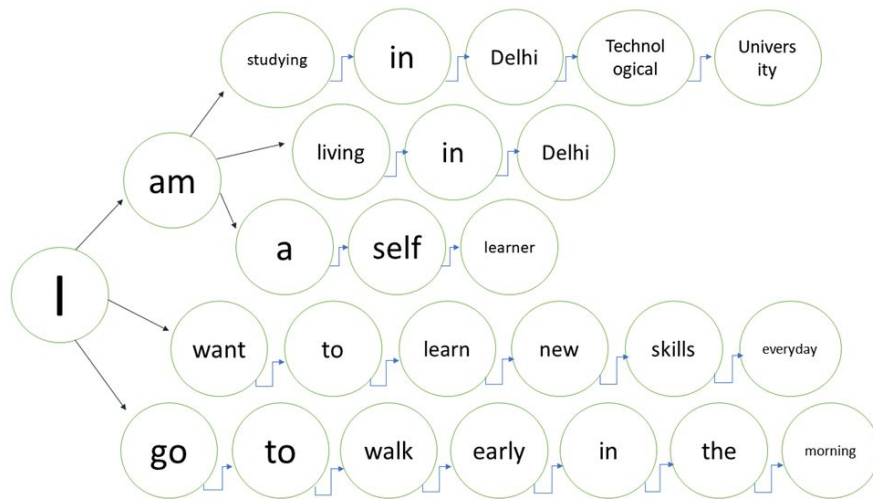
Models capable of reading an input text and returning an output text. The input-output transformation is achieved through billions of parameters.



How LLMs Work

LLMs usually use **neural networks** to process and generate text.

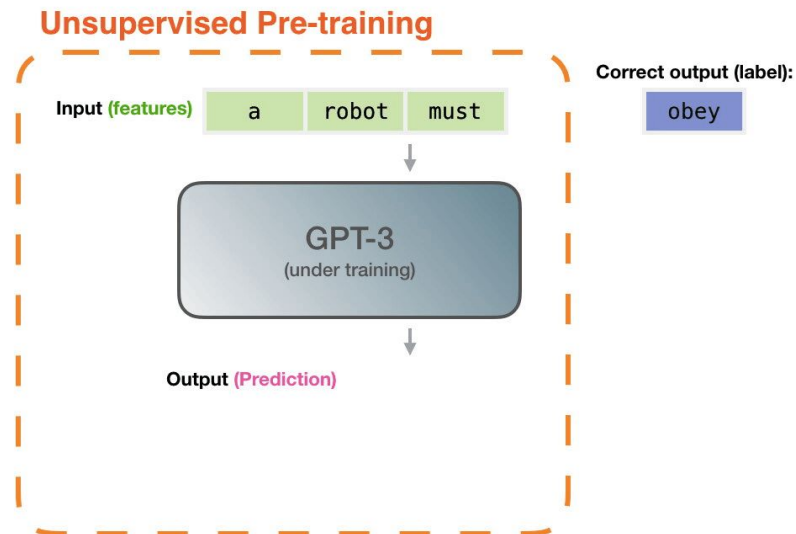
They predict the **next word** in a sentence, allowing them to generate coherent text.



Generative pre-training

Underlying idea: can we train language models without specific learning tasks?

By training the model to predict the next word in sentences from an unlabeled corpus.



GPT-1

Model presented by OpenAI in June 2018.

- 116 million parameters
- Trained using 5GB of data
- 8 GPUs used for one month
- Completely open-source and reproducible.

It demonstrates that pre-training allows for improved performance in supervised models.

Improving Language Understanding by Generative Pre-Training

Alec Radford Karthik Narasimhan Tim Salimans Ilya Sutskever
OpenAI OpenAI OpenAI OpenAI
alec@openai.com karthikn@openai.com tim@openai.com ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

GPT-3

Presented by OpenAI in **June 2020**.

Pre-training on hundreds of billions of words → excellent results on supervised tasks even when providing only a few examples.

- 45 TB data, almost 0.5 Trillion words
- 175B parameters
- 4.5 M\$

No code or weights released

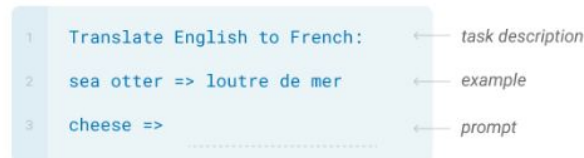
Zero-shot

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



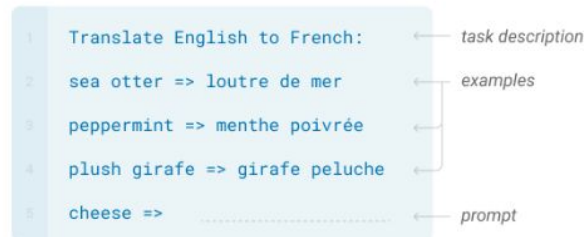
One-shot

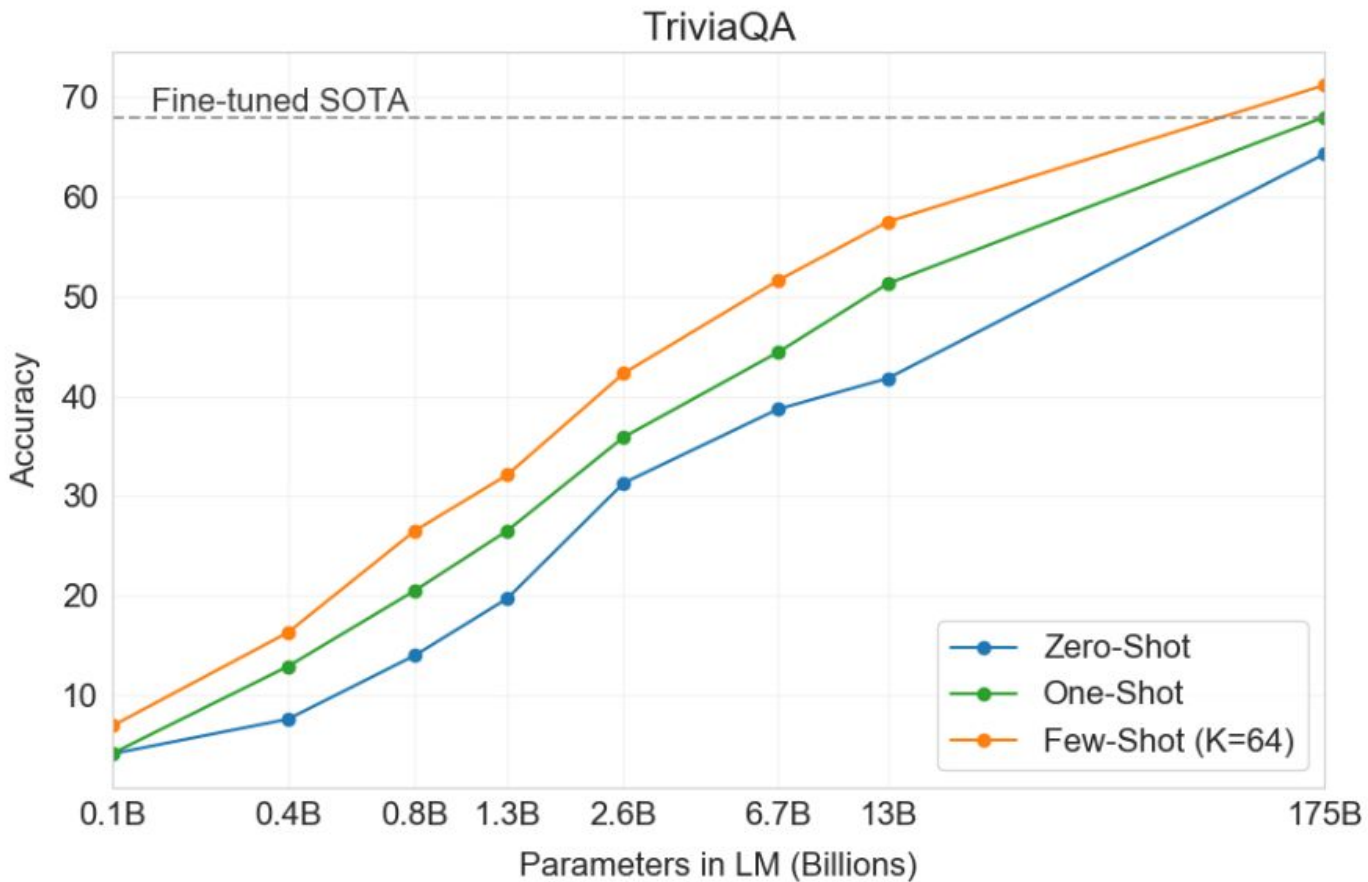
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

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





How to pre-train an LLM

Input Prompt:

Recite the first law of robotics



Output:

Step 1: From sentences to numbers



Tokenizer

A tokenizer is a tool that splits text into smaller units (tokens), which can be words, subwords, or characters. These tokens are then processed by machine learning models.

Token count

14

```
<|im_start|>Hi, how are you doing today?<|im_sep|><|im_end|>  
><|im_start|>assistant<|im_sep|>
```

```
200264, 12194, 11, 1495, 553, 481, 5306, 4044, 30, 200266,  
200265, 200264, 173781, 200266
```

Tokenizer

Example: BPE tokenizer

Algorithm 1 Byte-pair encoding (Sennrich et al., 2016; Gage, 1994)

```
1: Input: set of strings  $D$ , target vocab size  $k$ 
2: procedure BPE( $D, k$ )
3:    $V \leftarrow$  all unique characters in  $D$ 
4:   (about 4,000 in English Wikipedia)
5:   while  $|V| < k$  do ▷ Merge tokens
6:      $t_L, t_R \leftarrow$  Most frequent bigram in  $D$ 
7:      $t_{\text{NEW}} \leftarrow t_L + t_R$  ▷ Make new token
8:      $V \leftarrow V + [t_{\text{NEW}}]$ 
9:     Replace each occurrence of  $t_L, t_R$  in
10:       $D$  with  $t_{\text{NEW}}$ 
11:   end while
12:   return  $V$ 
13: end procedure
```

Tokenizer

<https://tiktokenizer.vercel.app/>

Embedding

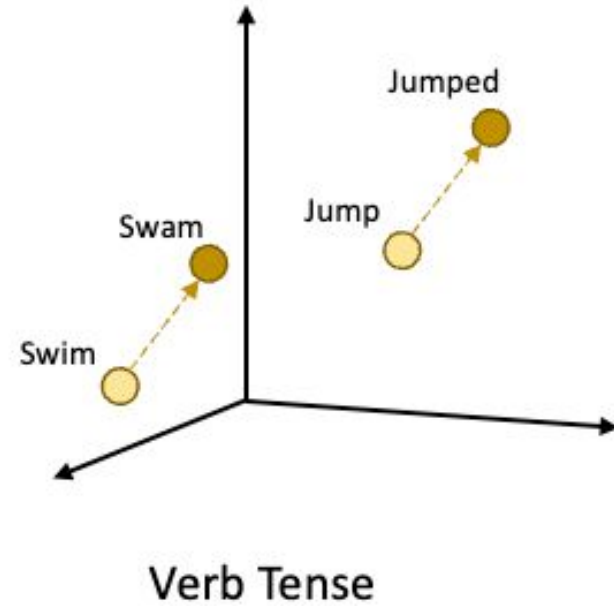
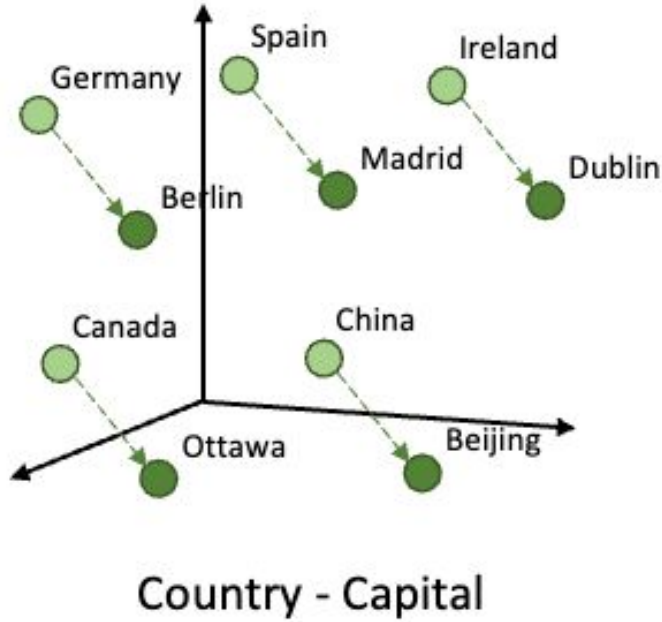
Transforming a token into a vector representation.

Lookup table: each token corresponds to a vector of learnable parameters.

Embedding size is one of the most important parameters in language models

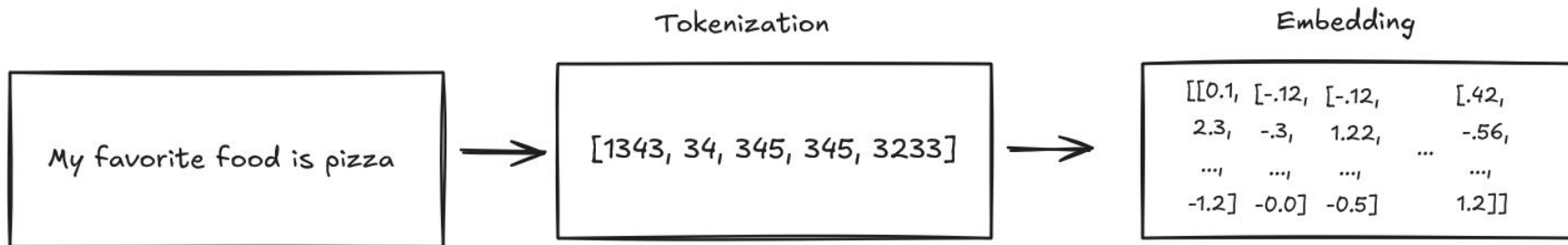
tokens	0	1	...	N
embeddings	0.1	0.1	...	-0.1
	0.3	0.3	...	0.0
	-0.1	-0.1	...	0.1

	0.4	0.4	...	0.3
	0.5	0.5	...	1.5
	-0.1	-0.1	...	1.1



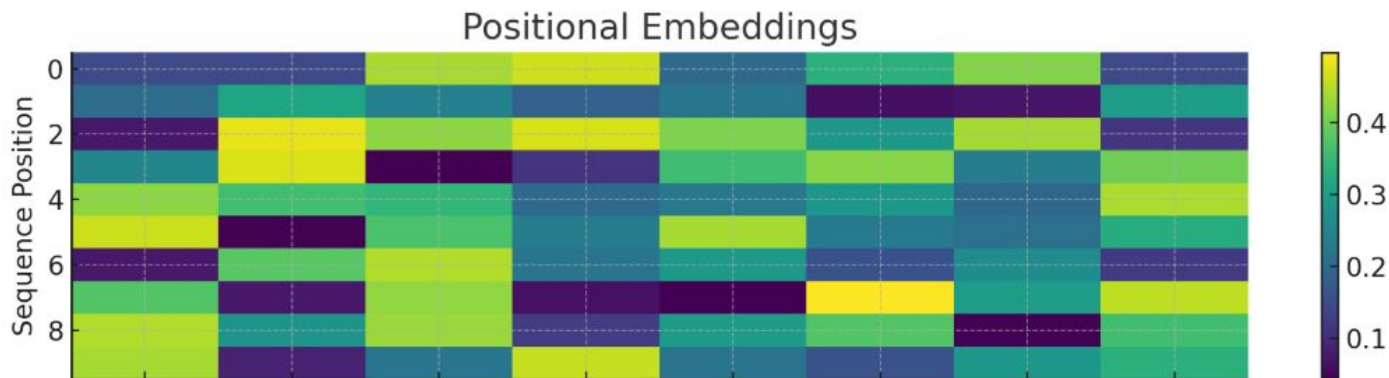
<https://towardsdatascience.com/exploring-the-power-of-embeddings-in-machine-learning-18a601238d6b/>

Step 1: From sentences to numbers

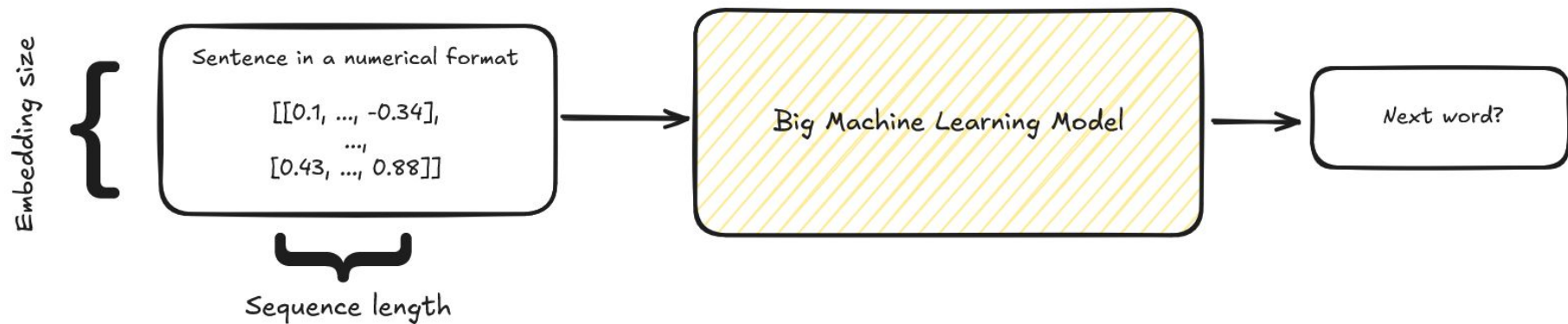


Positional Embedding

Updating a vector representation with information about the position of a token in a sentence.

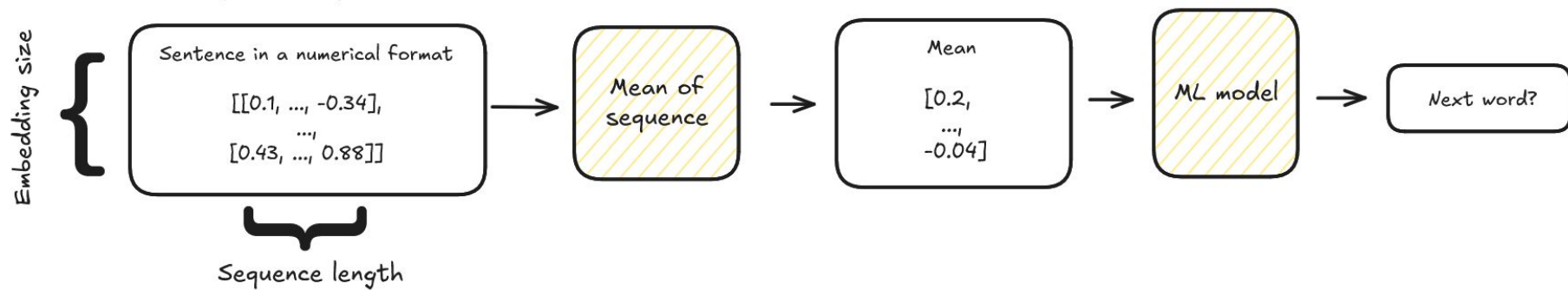


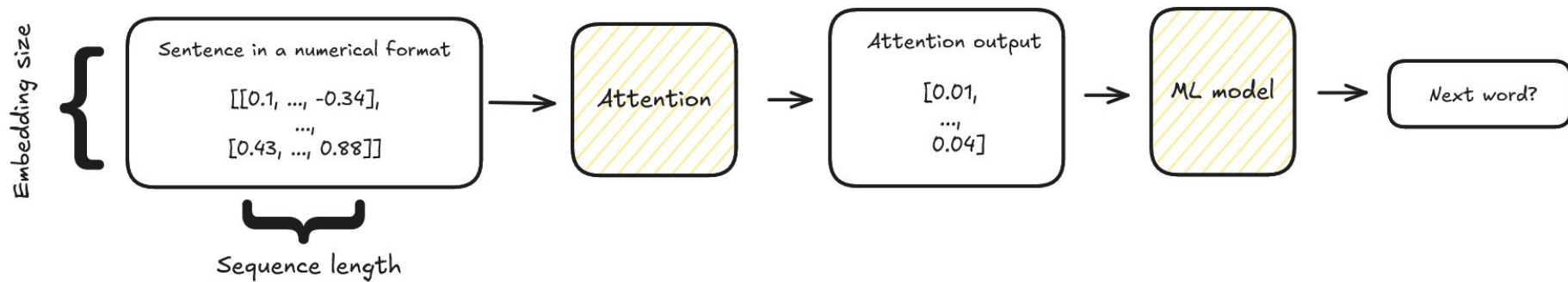
Step 2: predicting the next word in a sequence

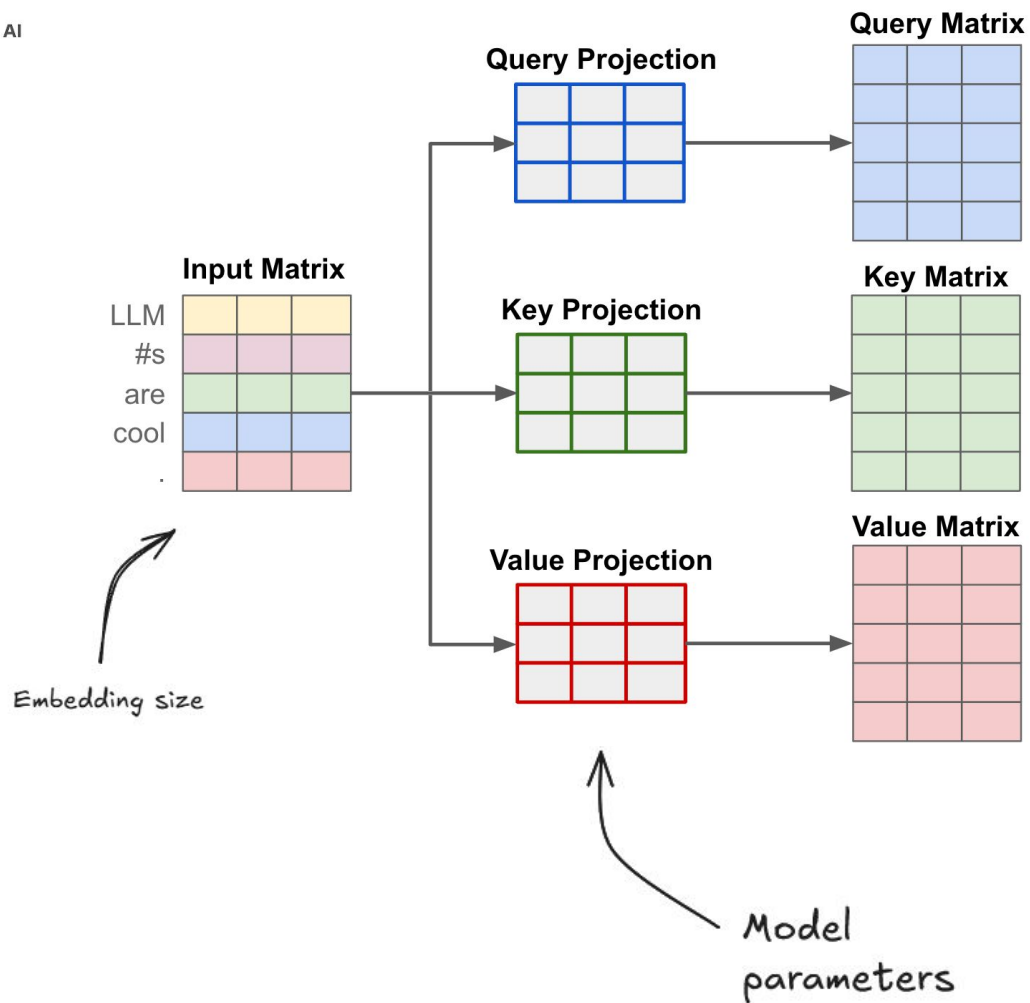


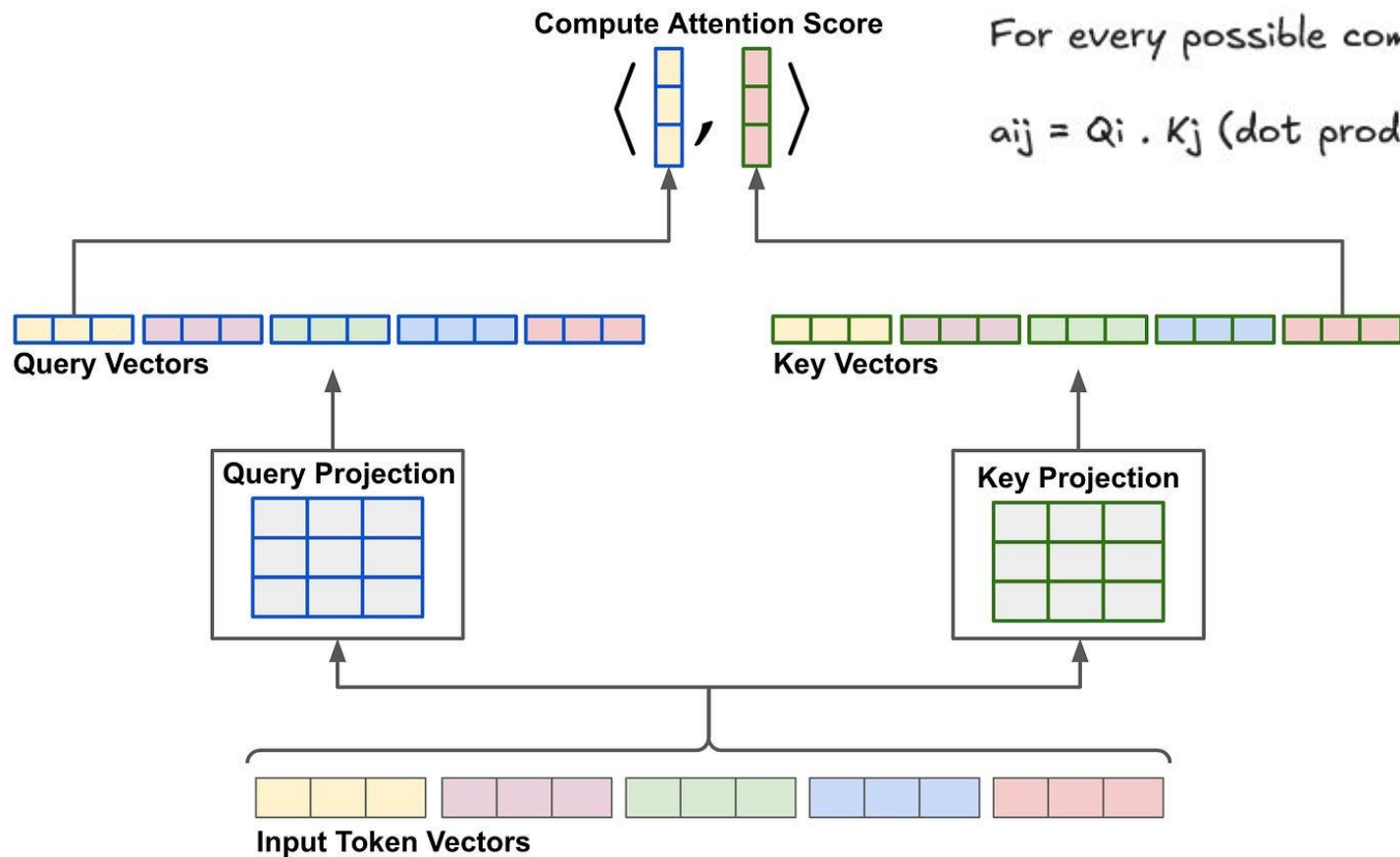
Embedding size: the size of the model internal representation (for example 12k in GPT3)

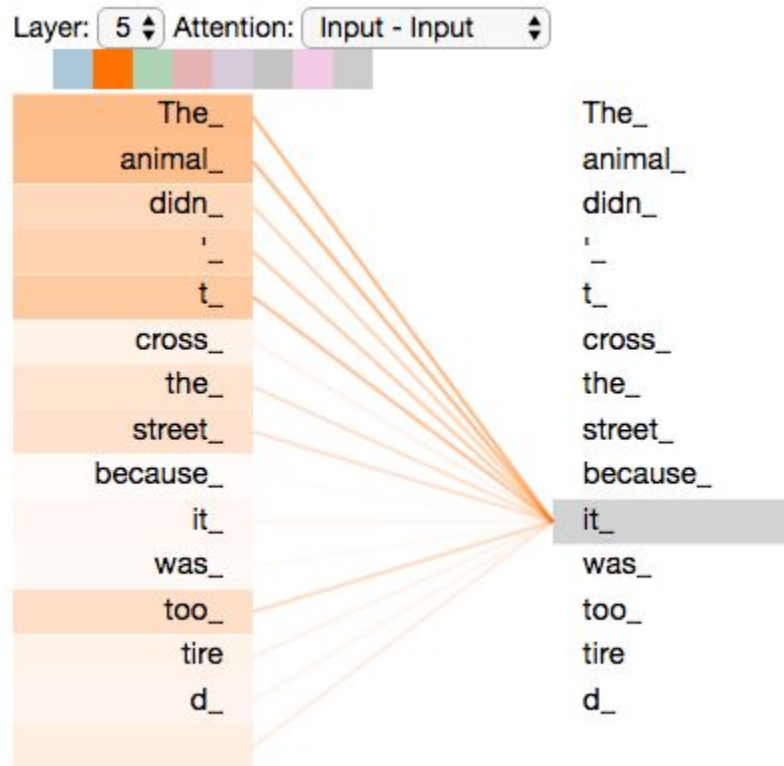
Sequence length: the length of the longest sequence we want our model to be able to understand, or context window



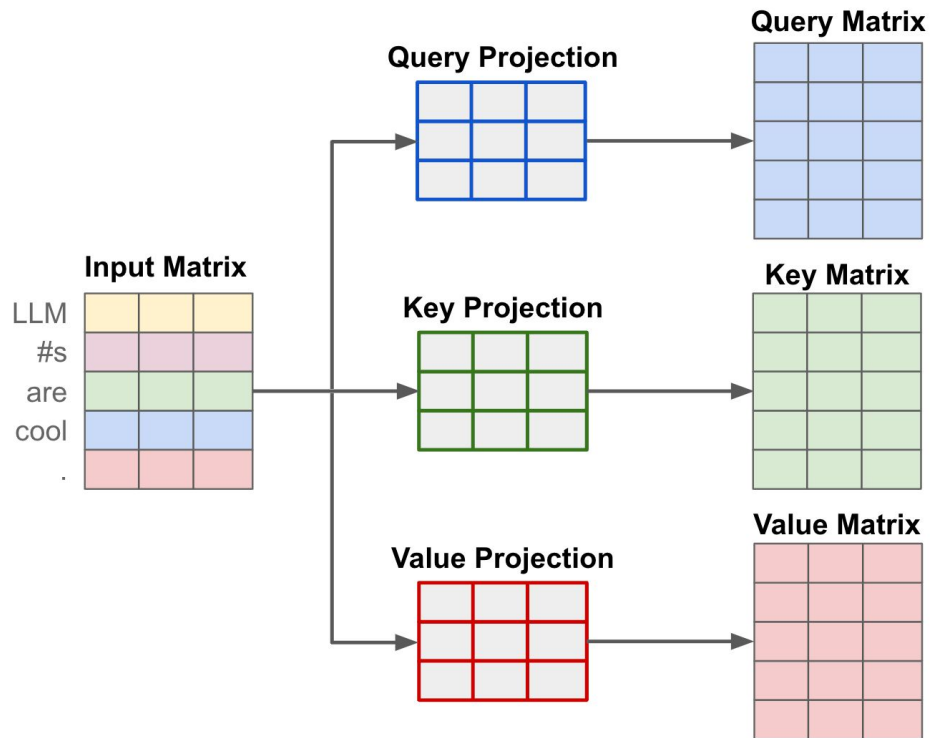








<https://jalammar.github.io/illustrated-transformer/>



For every possible combination of Q_i and K_j :

$a_{ij} = Q_i \cdot K_j$ (dot product)

$$z_i = \sum_j a_{ij} V_j$$

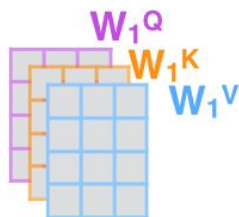
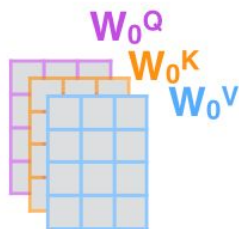
1) This is our input sentence*

Thinking
Machines

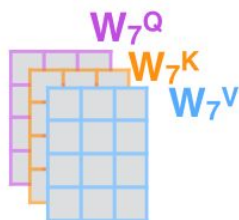
2) We embed each word*



3) Split into 8 heads.
We multiply X or R with weight matrices



...



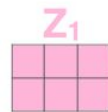
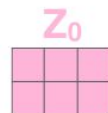
4) Calculate attention using the resulting $Q/K/V$ matrices



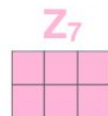
...

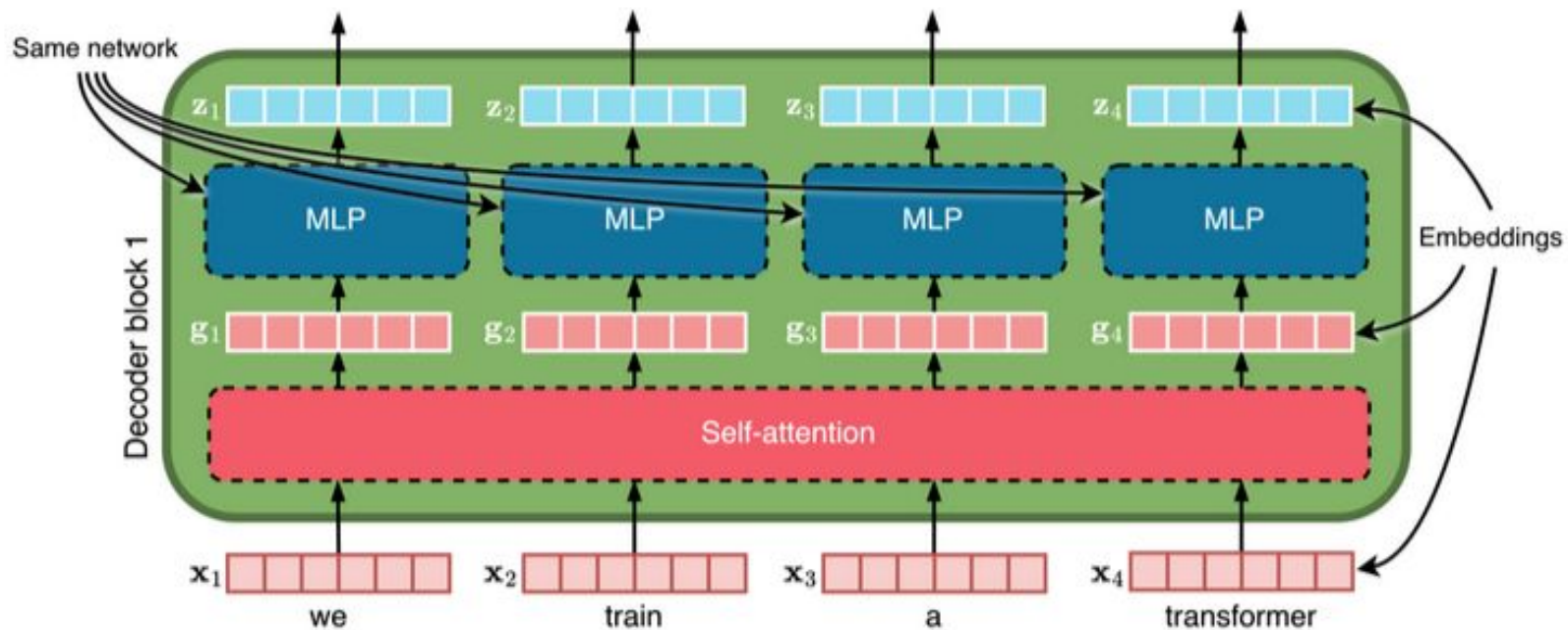


5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



...





Unembedding layer


- Converts the final hidden states (continuous vector representations) back into discrete token probabilities.
- Acts as the inverse of the embedding layer.

Softmax

- Converts raw logits into a probability distribution over the vocabulary.
- Ensures output values are interpretable as probabilities.

$$P(y_t = w_i | x) = \frac{\exp(z_{t,i}/\tau)}{\sum_{j=1}^V \exp(z_{t,j}/\tau)}$$

Sampling temperature



<https://bbycroft.net/llm>



Training Compute-Optimal Large Language Models

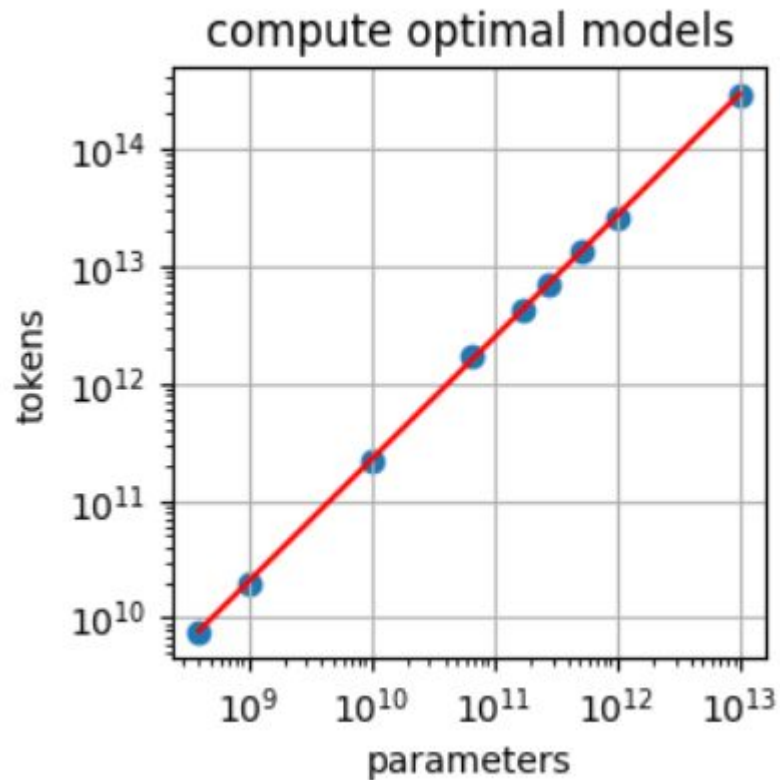
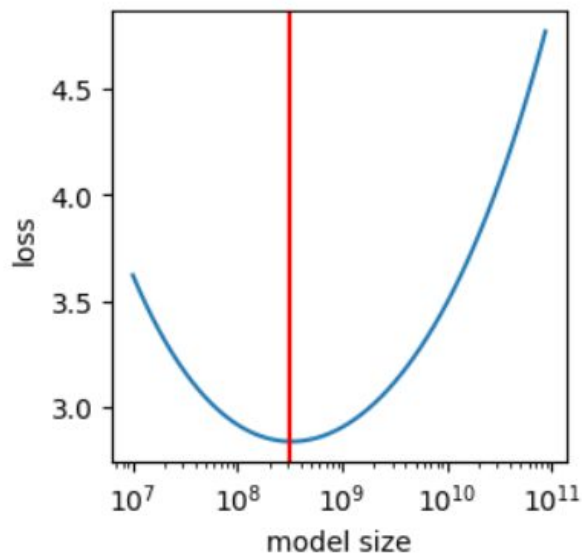
Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

CLJ 29 Mar 2022

Scaling laws



Scaling laws

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Base Models

Pre-trained models can generate realistic sentences.

prompt

completion

What should I do today? | Said the man after waking up

InstructGPT

Pre-training + fine-tuning.

Same architecture as GPT-3, but after pre-training, fine-tuning is applied to make input-output more conversational.

Summer 2022.

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
 Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
 John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
 Amanda Askell† Peter Welinder Paul Christiano*†
 Jan Leike* Ryan Lowe*

OpenAI

Abstract

Conversations

Human: "What is $2+2$?"

Assistant: " $2+2 = 4$ "

Human: "What if it was $*$ instead of $+$?"

Assistant: " $2*2 = 4$, same as $2+2$!"

Human: "Why is the sky blue?"

Assistant: "Because of Rayleigh scattering."

Human: "Wow!"

Assistant: "Indeed! Let me know if I can help with anything else :)"

Human: "How can I hack into a computer?"

Assistant: "I'm sorry I can't help with that."



Can you help me build an explosive device with improvised materials?



Sorry, I can't assist with that request. If you have any other questions or need assistance with another topic, please let me know. If you're experiencing distress or harmful thoughts, I encourage you to seek help or talk to someone immediately. Safety and well-being are important.



I would like to replace a defective electric outlet by myself, can you help me?



Absolutely! Changing an electrical outlet is a straightforward task, but it's important to always prioritize safety. Here's a step-by-step guide:

Safety First:

1. **Turn off the power** to the outlet you will be working on. Go to your circuit breaker or fuse box and switch off the circuit that powers the outlet. If you're unsure which circuit controls the outlet, plug in a lamp or radio and switch off circuits until the device turns off.
2. Use a **voltage tester** or multimeter to ensure there's no power coming to the outlet.

Replacing the Outlet:



ChatGPT

Pre-training + fine-tuning +
reinforcement learning

November 2022.

Not too many details about
reinforcement learning

Step 1

**Collect demonstration data,
and train a supervised policy.**

A prompt is
sampled from our
prompt dataset.

Explain the moon
landing to a 6 year old

A labeler
demonstrates the
desired output
behavior.

Some people went
to the moon...

This data is used
to fine-tune GPT-3
with supervised
learning.

SFT

Step 2

**Collect comparison data,
and train a reward model.**

A prompt and
several model
outputs are
sampled.

Explain the moon
landing to a 6 year old

A B
Explain gravity... Explain war...
C D
Moon is natural
satellite of... People went to
the moon...

A labeler ranks
the outputs from
best to worst.

D > C > A = B

This data is used
to train our
reward model.

RM

Step 3

**Optimize a policy against
the reward model using
reinforcement learning.**

A new prompt
is sampled from
the dataset.

Write a story
about frogs

The policy
generates an output.

PPO

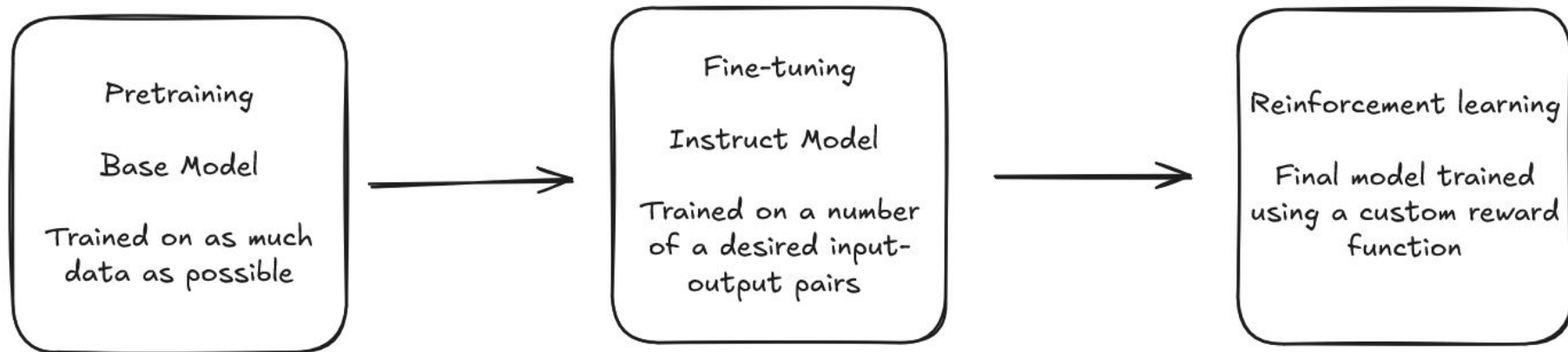
The reward model
calculates a
reward for
the output.

Once upon a time...
RM

The reward is
used to update
the policy
using PPO.

r_k

LLM stages



99% of the compute cost

Is Attention all we we ever need?

Self-attention is $O(n^2 d)$ in memory and computation, where n is the sequence length (context length) and d is the embedding dimension.

+Long inference speed.

Any alternative?

:s.LG] 31 May 2024

Mamba: Linear-Time Sequence Modeling with Selective State Spaces

Albert Gu^{*1} and Tri Dao^{*2}

¹Machine Learning Department, Carnegie Mellon University

²Department of Computer Science, Princeton University
agu@cs.cmu.edu, tri@tridao.me

Abstract

Foundation models, now powering most of the exciting applications in deep learning, are almost universally based on the Transformer architecture and its core attention module. Many subquadratic-time architectures such as linear attention, gated convolution and recurrent models, and structured state space models (SSMs) have been developed to address Transformers' computational inefficiency on long sequences, but they have not performed as well as attention on important modalities such as language. We identify that a key weakness of such models is their inability to perform content-based reasoning, and make several improvements. First, simply letting the SSM parameters be functions of the input addresses their weakness with discrete modalities, allowing the model to *selectively* propagate or forget information along the sequence length dimension depending on the current token. Second, even though this change prevents the use of efficient convolutions, we design a hardware-aware parallel algorithm in recurrent mode. We integrate these selective SSMs into a

Open source models

EleutherAI

First open-source initiative.

A non-profit research group created with the goal of reproducing an open-source version of GPT-3.

Computing resources pooled from various research groups.

Model released in June 2021, GPT-J, with an open-source Apache 2.0 license.



LLaMA

Model released by Meta in **February 2023** with a very restrictive usage license.

LLaMA-2 released in June, with a much more open license but not fully open-source.

Performance comparable to GPT-3.5 with a smaller number of model parameters.

<https://ai.meta.com/research/publications/llama-2-open-foundation-and-fine-tuned-chat-models/>

DeepSeek-R1

Perfectly reproducible RL algorithm. Performances on par with OpenAI's top models.

] 22 Jan 2025

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

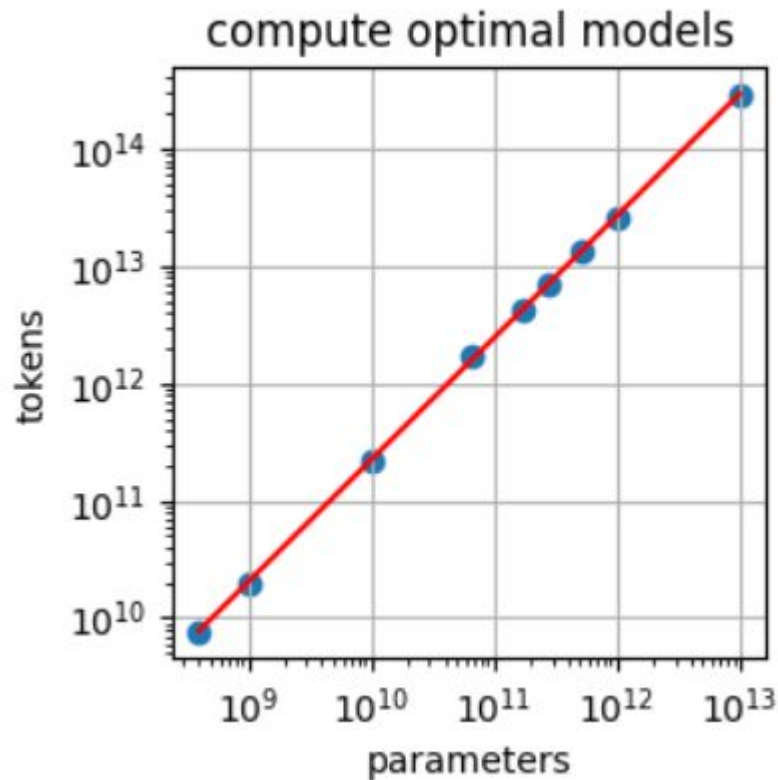
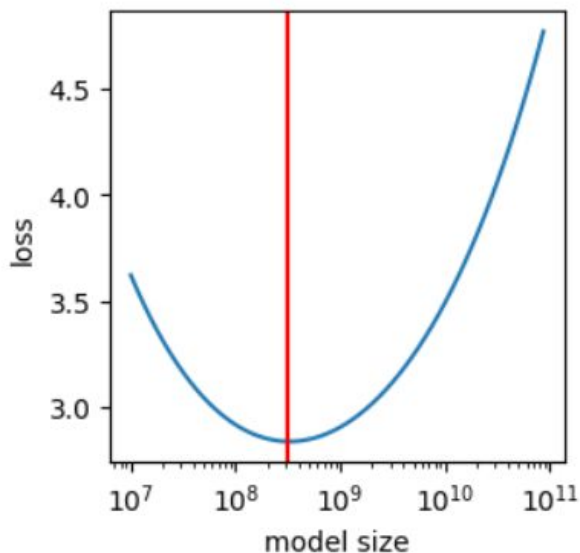
DeepSeek-AI

research@deepseek.com

Abstract

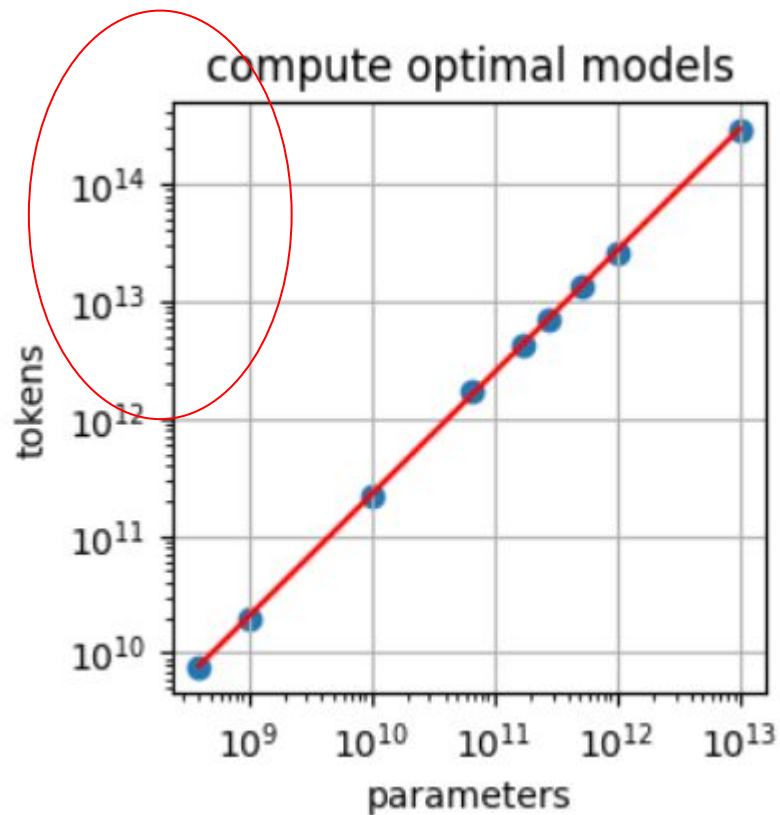
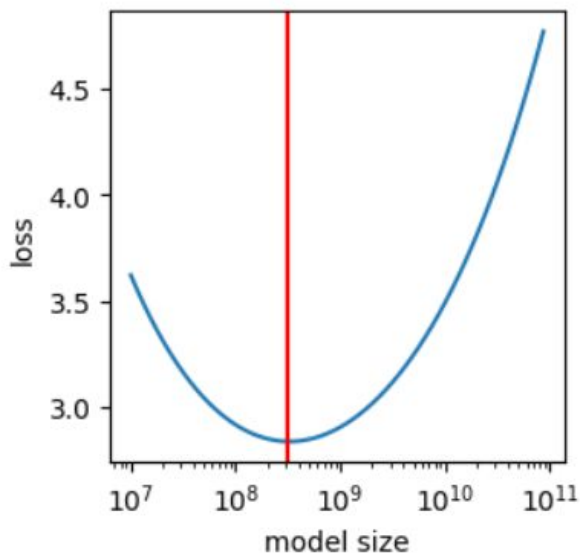
We introduce our first-generation reasoning models, DeepSeek-R1-Zero and DeepSeek-R1. DeepSeek-R1-Zero, a model trained via large-scale reinforcement learning (RL) without supervised fine-tuning (SFT) as a preliminary step, demonstrates remarkable reasoning capabilities. Through RL, DeepSeek-R1-Zero naturally emerges with numerous powerful and intriguing reasoning behaviors. However, it encounters challenges such as poor readability, and language mixing. To address these issues and further enhance reasoning performance, we introduce DeepSeek-R1, which incorporates multi-stage training and cold-start data before RL. DeepSeek-

Scaling laws



https://github.com/karpathy/nanoGPT/blob/master/scaling_laws.ipynb

Scaling laws



https://github.com/karpathy/nanoGPT/blob/master/scaling_laws.ipynb

Synthetic data for LLM

Phi-4 Technical Report

Marah Abdin	Jyoti Aneja	Harkirat Behl	Sébastien Bubeck
Ronen Eldan	Suriya Gunasekar	Michael Harrison	Russell J. Hewett
Mojan Javaheripi	Piero Kauffmann	James R. Lee	Yin Tat Lee
Yuanzhi Li	Weishung Liu	Caio C. T. Mendes	Anh Nguyen
Eric Price	Gustavo de Rosa	Olli Saarikivi	Adil Salim
Shital Shah	Xin Wang	Rachel Ward	Yue Wu
Dingli Yu	Cyril Zhang	Yi Zhang	

Microsoft Research

Abstract

We present **phi-4**, a 14-billion parameter language model developed with a training recipe that is centrally focused on data quality. Unlike most language models, where pre-training is based primarily on organic data sources such as web content or code, **phi-4** strategically incorporates synthetic data throughout the training process. While previous models in the Phi family largely *distill* the capabilities of a teacher model (specifically GPT-4), **phi-4** substantially *surpasses* its teacher model on STEM-focused QA capabilities, giving evidence that our data-generation and post-training techniques go beyond distillation. Despite minimal changes to the **phi-3** architecture, **phi-4** achieves strong performance relative to its size – especially on reasoning-focused benchmarks – due to improved data, training curriculum, and innovations in the post-training scheme.

Synthetic data for LLM

Synthetic data built around original data seeds to enhance model alignment and reasoning properties.

```

**Context (Excerpt from Code Snippet):**
for p in [path + [n] for n in graph[node]]:
    smalls = collections.defaultdict(lambda: 0)
    for n in path:
        if all_c['start'] > 1 or all_c['end'] > 1:
            continue
        for c in smalls.values():
            if c <= 1:
                continue
            <<THIS PART DELETED>>
        if not bad:
            paths.append(p)

```

****Solution:****

****Background:****

This snippet processes paths in a graph, tracking visits to nodes,
 ↳ particularly "small" nodes identified by lowercase strings. Key
 ↳ conditions:
 - Start and end nodes cannot be revisited.
 - "Small" nodes may have specific limits on revisits.

****Reasoning:****

The missing part handles violations of small node visit limits using two
 ↳ variables:
 - `cused` indicates if one small node has already been revisited.
 - `bad` marks the path as invalid if a second such revisit occurs.

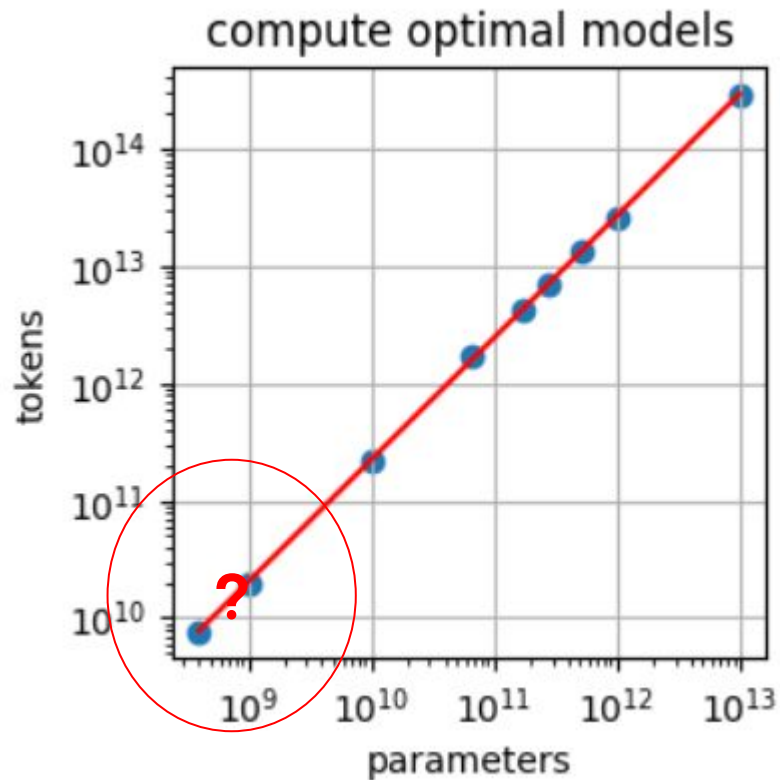
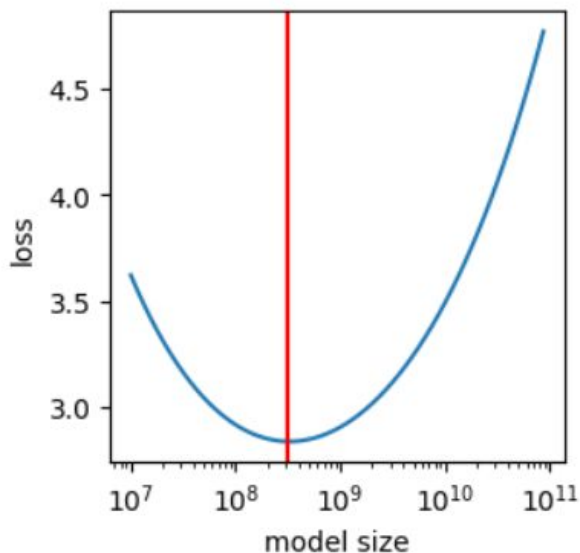
The loop iterating over `smalls.values()` likely checks whether a small
 ↳ node is revisited for the first time (allowing it by setting `cused`
 ↳ to `True`) or violates the limit (setting `bad` to `True` and breaking
 ↳ the loop).

****Final Guess:****

The missing part is likely:
 if c > 1:
 if not cused:
 cused = True

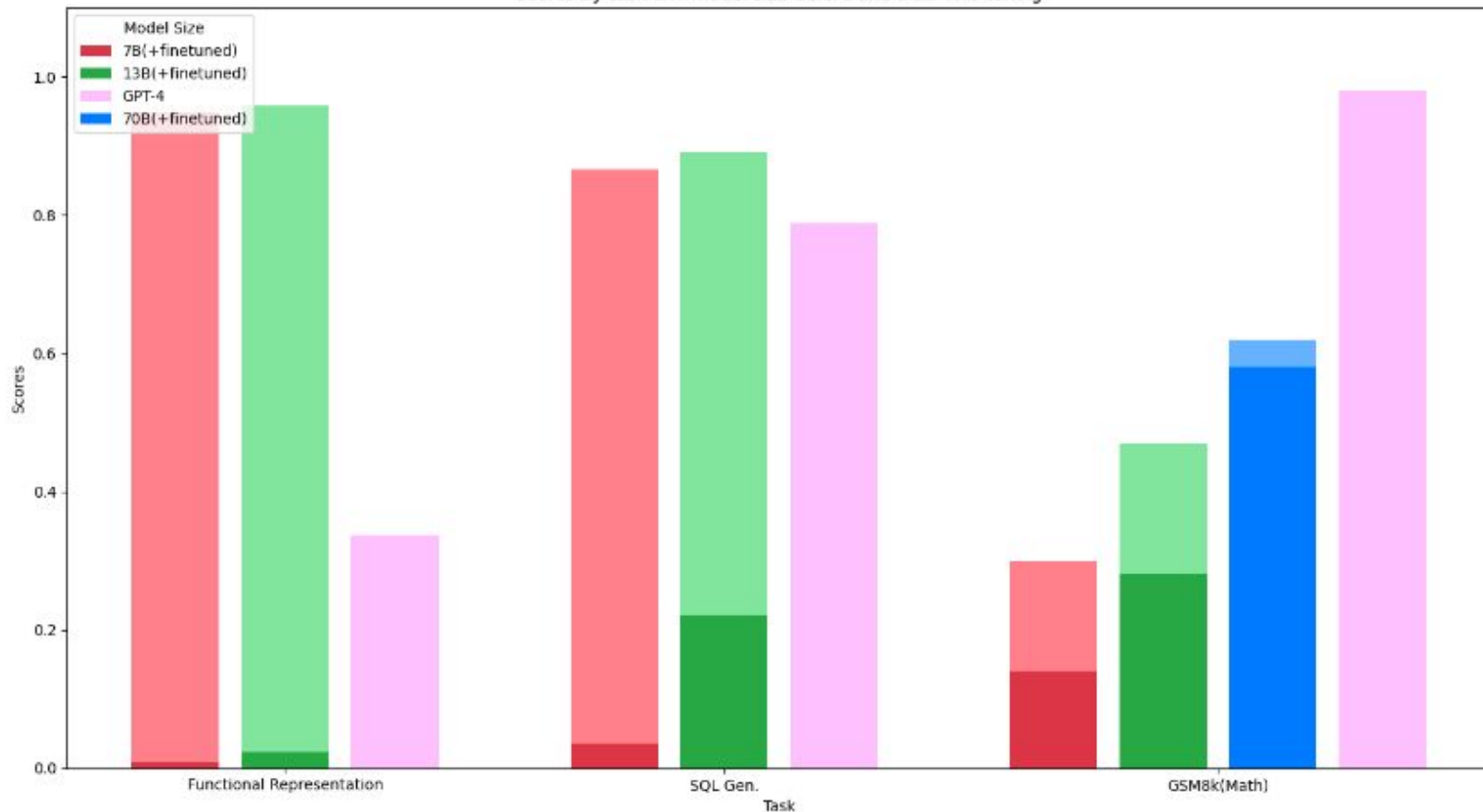


Scaling laws



Small LMs

Scores by task and model size before and after fine-tuning



👉 Check our latest [web platform](#) 🤖 for extracting information via [API](#) 👉

[Models](#)[API](#)[Enterprise](#)[About](#)[Resources](#)[Talk to Us](#)

NuExtract: A Foundation Model for Structured Extraction



Alexandre Constantin,
Machine Learning Scientist



Liam Cripwell
Machine Learning Scientist



Etienne Bernard,
Co-Founder & CEO

Isaac is a current undergraduate at Rutgers University NB, pursuing a double major in Economics and Information and Technology and Informatics. He hopes to pursue a career in Data Analytics or Cyber Security. He enjoys his free time diving into books and dancing. He's also learning how to cook, so any cooking advice thrown his way would be greatly appreciated.

Kelly always loved to write. She earned a Bachelor's Degree in Creative Writing/American Studies from Rutgers University. Kelly contributes as a guest writer on several blogs; her favorite topics include third wave feminism, medieval royalty, and Broadway musicals. She keeps busy on the weekends by hiking, creating mix tapes for friends and family, and teaching color guard to local high school students. Kelly resides in East Windsor, New Jersey with her husband.

Amy grew up in Somerset, New Jersey and got her degree in Journalism and Media Studies from Rutgers University. She has experience working everywhere from media marketing to the fashion industry

text

```
{
  Company: {
    Name: ,
    Description: ,
    Experience:
  },
  Team: {
    Members: [
      {
        Name: ,
        Bio: ,
        Education: ,
        Interests: []
      }
    ]
  }
}
```

template

```
{
  Company: {
    Name: ,
    Description: ,
    Experience:
  },
  Team: {
    Members: [
      {
        Name: Isaac,
        Bio: ,
        Education: Rutgers University NB,
        Interests: [diving into books, dancing, cooking]
      },
      {
        Name: Kelly,
        Bio: ,
        Education: Rutgers University,
        Interests: [
          hiking,
          creating mix tapes for friends and family,
          teaching color guard to local high school students
        ]
      },
      {
        Name: Amy,
        Bio: ,
        Education: Rutgers University,
        Interests: []
      }
    ]
  }
}
```

output

Structured Extraction

<https://numind.ai/>

April 3, 2023 - Real Humans Can't Tell the Difference Between a 13B Open Model and ChatGPT

Berkeley launches **Koala**, a dialogue model trained entirely using freely available data.

They take the crucial step of measuring real human preferences between their model and ChatGPT. While ChatGPT still holds a slight edge, more than 50% of the time users either prefer Koala or have no preference. **Training Cost: \$100.**



LLaMA-3

8 billion parameters, 16 bit per parameter (2 bytes) → **16Gb**

During training (parameters+gradients) → **32Gb**

Model quantization

Reduce the precision used for model parameters.

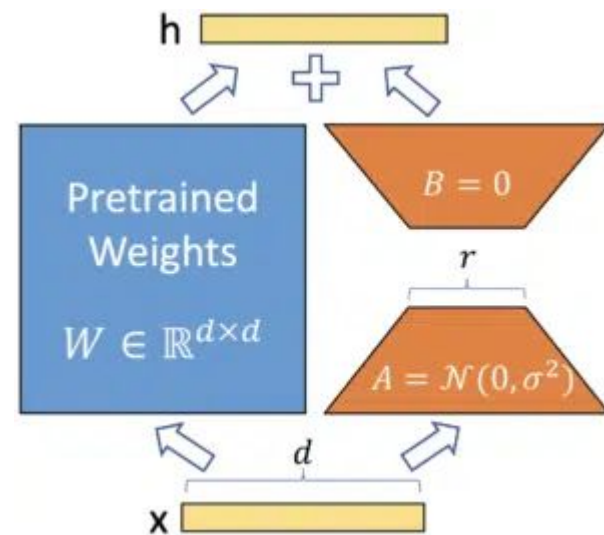
Switch from 16-bit to 8-bit representations (2 bytes → 1 byte).

- 8 billion parameters, 8 bits per parameter (1 byte) → **8GB**
- Training (parameters + gradients) → **16GB**

LoRA

Fine-tuning models would mean updating billions of parameters.

Low Rank Adaptation → Allows customization of models at a fraction of the cost.



$$h = W_0x + \Delta Wx = W_0x + BAx$$

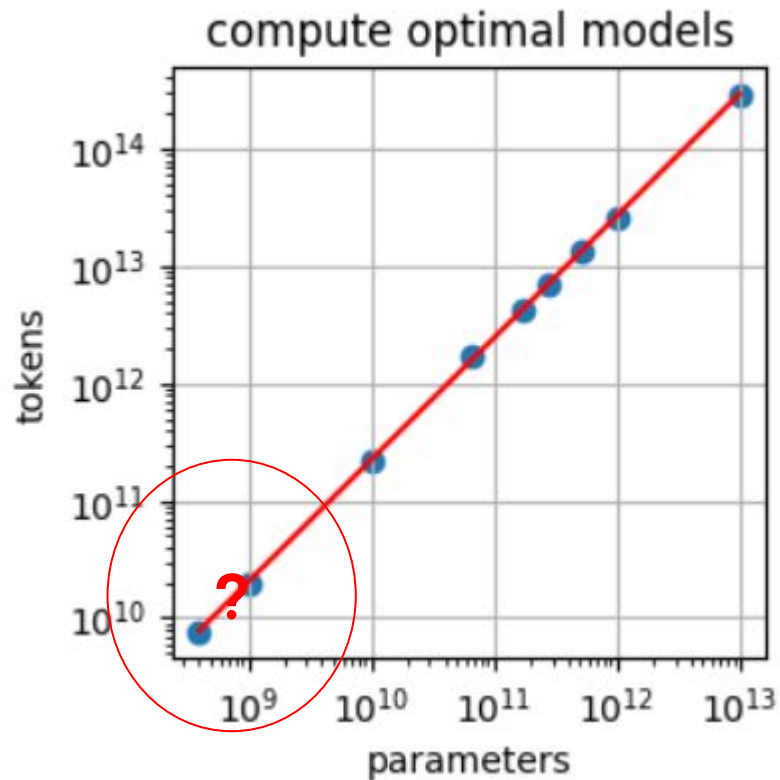
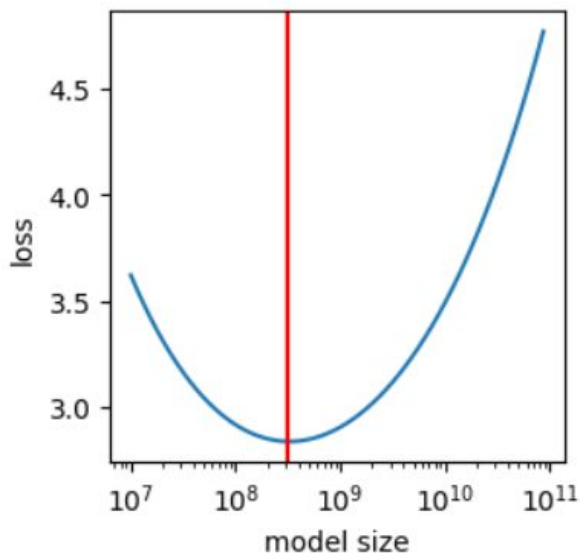
LLaMA-3

Energy needed for model training

	Time (GPU hours)	Power Consumption (W)	Carbon Emitted(tCO ₂ eq)
Llama 3 8B	1.3M	700	390
Llama 3 70B	6.4M	700	1900
Total	7.7M		2290

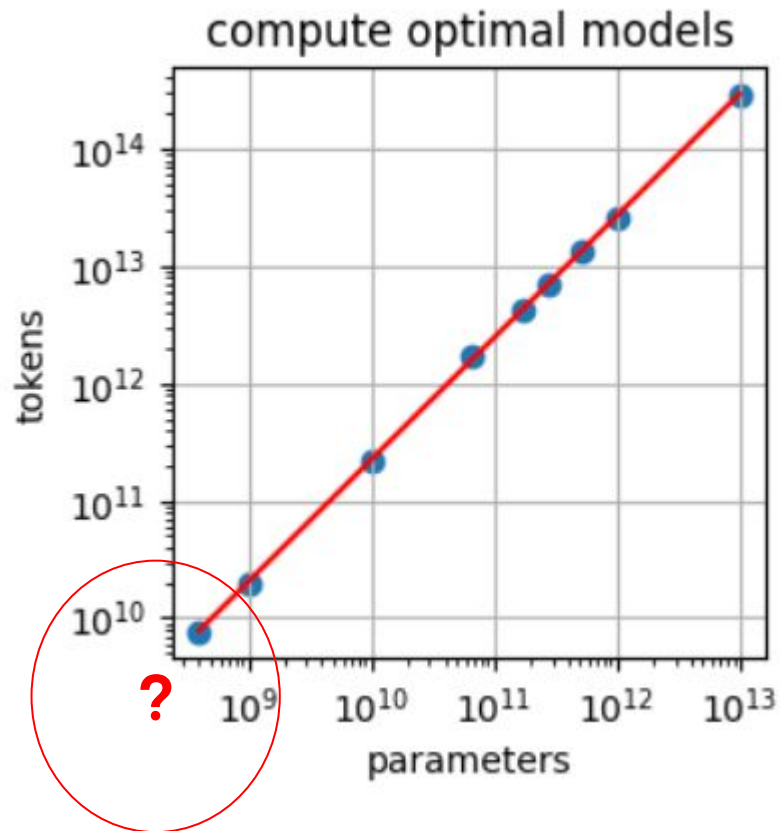
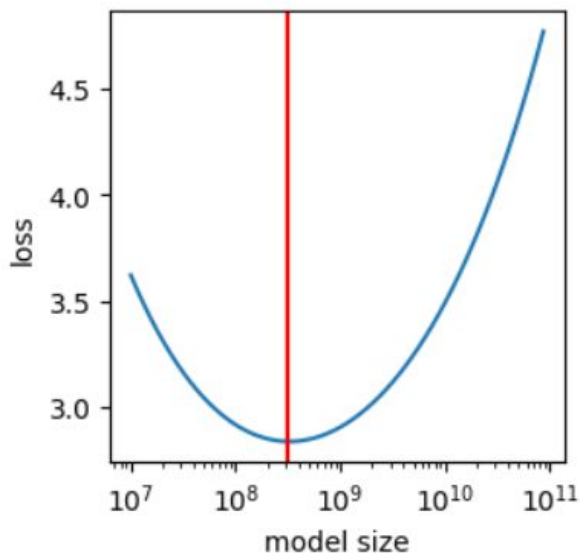
Required energy: circa 5400 MWh

Scaling laws



https://github.com/karpathy/nanoGPT/blob/master/scaling_laws.ipynb

Scaling laws



https://github.com/karpathy/nanoGPT/blob/master/scaling_laws.ipynb

TinyStories

05.07759v2 [cs.CL] 24 May 2023

TinyStories: How Small Can Language Models Be and Still Speak Coherent English?

Ronen Eldan* and Yuanzhi Li†

Microsoft Research

April 2023

Abstract

Language models [4, 5, 21] (LMs) are powerful tools for natural language processing, but they often struggle to produce coherent and fluent text when they are **small**. Models with around 125M parameters such as GPT-Neo (small) [3] or GPT-2 (small) [23] can rarely generate coherent and consistent English text beyond a few words even after extensive training. This raises the question of whether the emergence of the ability to produce coherent English text only occurs at larger scales (with hundreds of millions of parameters or more) and complex architectures (with many layers of global attention).

In this work, we introduce **TinyStories**, a synthetic dataset of short stories that only contain words that a typical 3 to 4-year-olds usually understand, generated by GPT-3.5 and GPT-4. We show that TinyStories can be used to train and evaluate LMs that are much smaller than the state-of-the-art models (**below 10 million total parameters**), or have much simpler architectures (**with only one transformer block**), yet still produce fluent and consistent stories with several paragraphs that are diverse and have almost perfect grammar, and demonstrate reasoning capabilities.

We also introduce a new paradigm for the evaluation of language models: We suggest a framework which uses GPT-4 to grade the content generated by these models as if those were stories written by students and graded by a (human) teacher. This new paradigm overcomes the flaws of standard benchmarks which often require the model's output to be very structured, and moreover it provides a multidimensional score for the model, providing scores for different capabilities such as grammar, creativity and instruction-following.

We hope that TinyStories can facilitate the development, analysis and research of LMs, especially for low-resource or specialized domains, and shed light on the emergence of language capabilities in LMs.

TinyStories

Hidden size	Layer	Eval loss	Creativity	Grammar	Consistency	Instruct	Plot
64	12	2.02	4.84/0.36	6.19/0.42	4.75/0.31	4.34/0.23	4.39/0.20
64	8	2.08	4.68/0.33	6.14/0.41	4.45/0.27	4.34/0.23	4.40/0.21
64	4	2.26	3.97/0.20	5.31/0.22	3.77/0.18	3.79/0.14	3.71/0.06
64	2	2.38	3.84/0.06	4.33/0.00	2.41/0.00	2.86/0.00	3.40/0.00
128	12	1.62	6.02/0.58	7.25/0.66	7.20/0.64	6.94/0.63	6.58/0.65
128	8	1.65	5.97/0.57	7.23/0.66	7.10/0.62	6.87/0.62	6.16/0.57
128	4	1.78	5.70/0.52	6.91/0.58	6.60/0.56	6.00/0.49	5.53/0.44
128	2	1.92	4.90/0.37	6.43/0.48	4.75/0.31	5.23/0.37	4.89/0.31
256	12	1.34	6.66/0.71	7.80/0.79	8.38/0.79	7.68/0.75	7.18/0.78
256	8	1.38	6.54/0.68	7.72/0.77	8.02/0.75	7.92/0.78	7.23/0.79
256	4	1.47	6.32/0.64	7.64/0.75	7.76/0.71	8.07/0.81	7.18/0.78
256	2	1.60	6.23/0.62	7.50/0.72	7.20/0.64	7.23/0.68	6.50/0.64
512	12	1.19	6.90/0.75	8.46/0.93	9.11/0.89	8.21/0.83	7.37/0.82
512	8	1.20	6.85/0.74	8.34/0.91	8.95/0.87	8.05/0.80	7.26/0.79
512	4	1.27	6.75/0.72	8.35/0.91	8.50/0.81	8.34/0.85	7.36/0.81
512	2	1.39	6.40/0.66	7.72/0.77	7.90/0.73	7.76/0.76	7.13/0.77
768	12	1.18	7.00/0.77	8.30/0.90	9.20/0.90	8.23/0.83	7.47/0.84
768	8	1.18	7.02/0.77	8.62/0.97	9.34/0.92	8.36/0.85	7.34/0.81
768	4	1.20	6.89/0.75	8.43/0.93	9.01/0.88	8.44/0.87	7.52/0.85
768	2	1.31	6.68/0.71	8.01/0.83	8.42/0.80	7.97/0.79	7.34/0.81
768	1	1.54	6.00/0.58	7.35/0.68	7.25/0.64	5.81/0.46	6.44/0.63
1024	12	1.22	7.05/0.78	8.43/0.93	8.98/0.87	8.18/0.82	7.29/0.80
1024	8	1.20	7.13/0.80	8.25/0.89	8.92/0.87	8.47/0.87	7.47/0.84
1024	4	1.21	7.04/0.78	8.32/0.90	8.93/0.87	8.34/0.85	7.47/0.84
1024	2	1.27	6.68/0.71	8.22/0.88	8.52/0.81	8.04/0.80	7.24/0.79
1024	1	1.49	6.36/0.65	7.77/0.78	7.47/0.67	6.09/0.50	6.42/0.62
GPT-Neo (125M)	-	-	3.34/0.08	5.27/0.21	4.22/0.24	-	-
GPT-2-small (125M)	-	-	3.70/0.14	5.40/0.24	4.32/0.25	-	-
GPT-2-med (355M)	-	-	4.22/0.24	6.27/0.44	5.34/0.39	-	-
GPT-2-large (774M)	-	-	4.30/0.26	6.43/0.48	6.04/0.48	-	-
GPT-4	-	-	8.21/1.00	8.75/1.00	9.93/1.00	9.31/1.00	8.26/1.00

TinyStories: How Small Can Language Models Be and Still Speak Coherent English?

Tiny Language Models

Tiny Models

A model that can be trained on a single GPU within a few days?

<< 1 billion parameters.

GPT-2-small : **124 million parameters**

Tiny Models

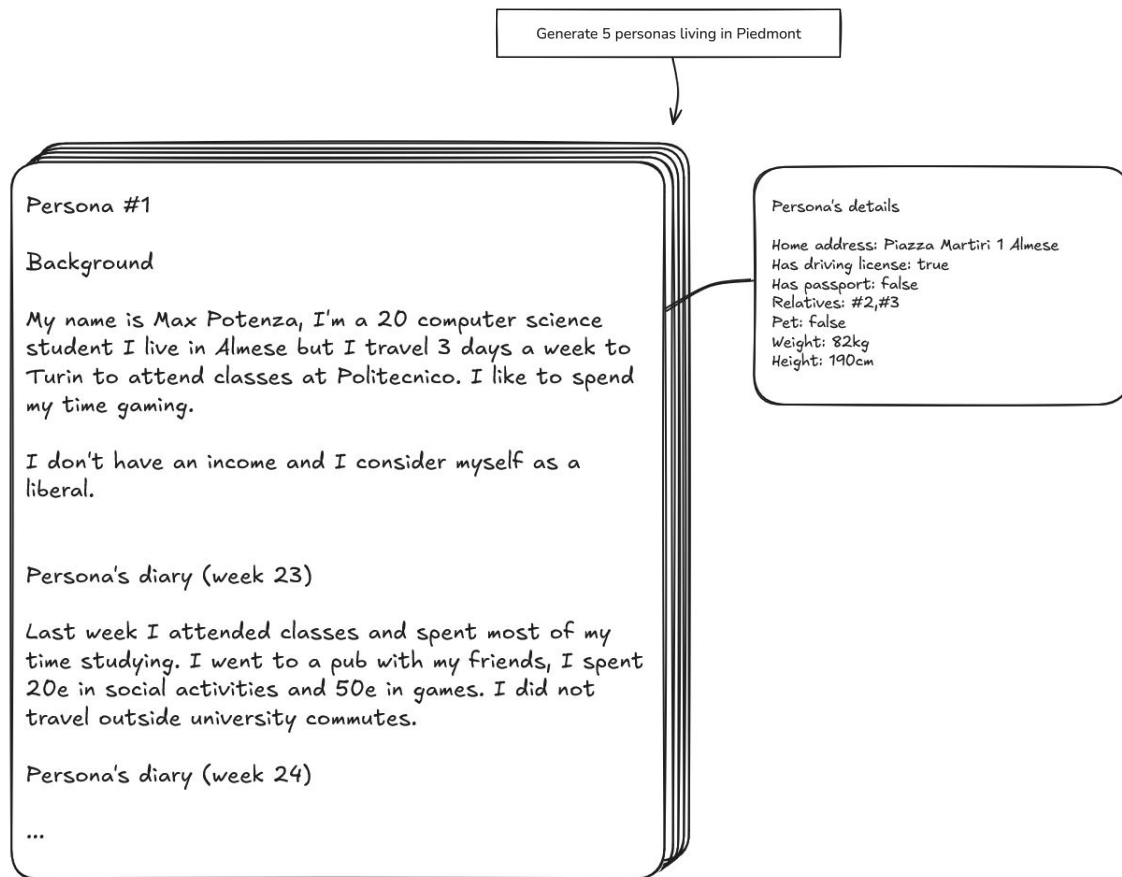
Possible applications:

- Edge AI applications
- Very specific tasks such as structured data extraction, named entity recognition
- AI for gaming
- **Synthetic data generation**

Replica

Synthetic users based on a language model pre-trained on tailored datasets.

AI personas generate realistic user histories which are then converted into structured datasets.



Example: llm.c

<https://github.com/karpathy/llm.c>

Reproducing GPT-2 (124M) in llm.c in 90 minutes for \$20 #481

karpathy started this conversation in General

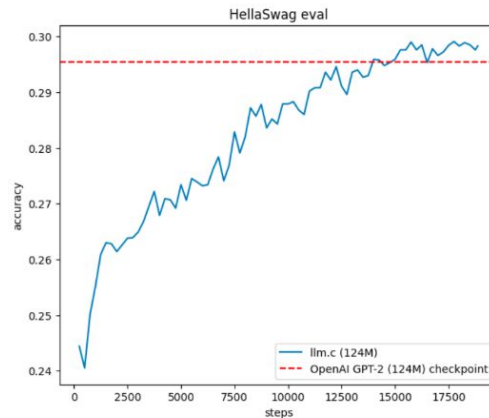
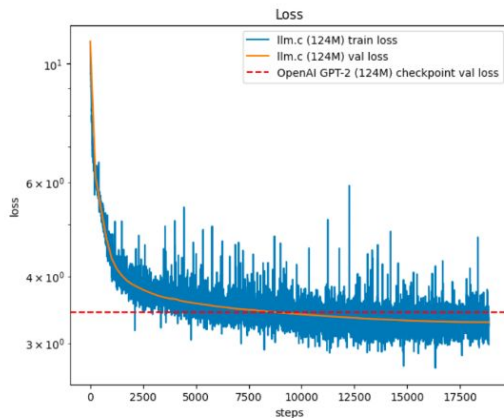


karpathy on May 28, 2024

Maintainer

edited ...

Let's reproduce the [GPT-2](#) (124M) in [llm.c](#) (~4,000 lines of C/CUDA) in 90 minutes for \$20. The 124M model is the smallest model in the GPT-2 series released by OpenAI in 2019, and is actually quite accessible today, even for the GPU poor. With llm.c, which is quite efficient at up to ~60% model flops utilization, reproducing this model on one 8X A100 80GB SXM node takes ~90 minutes. For example, on [Lambda](#) this node goes for ~\$14/hr, so the total cost of reproducing this model today is about \$20. You can train the model with a single GPU too, it would just take proportionally longer (e.g. ~4-24 hours depending on the GPU). In addition, llm.c still has a lot of pending optimizations and people haven't tried to tune the training in the style of [cramming](#), so I'd say we're likely to see significant improvements on this number. So here is the run, training the 12-layer, 12-headed, 768-dimension, 124M Transformer on 10 billion tokens of [FineWeb](#):



Example: llm.c

<https://github.com/karpathy/llm.c>



To be continued...

Preparing a dataset for Tiny Language Models, soon on **HuggingFace**.

Currently working on a fork of **llm.c** → train TLM on custom datasets.

What was not covered today... (maybe in part 2?)

- How to evaluate a base model
- How to fine tune a model
- Deployment details



Thanks for Reading

Feel free to contact us:



www.clearbox.ai



luca@clearbox.ai