

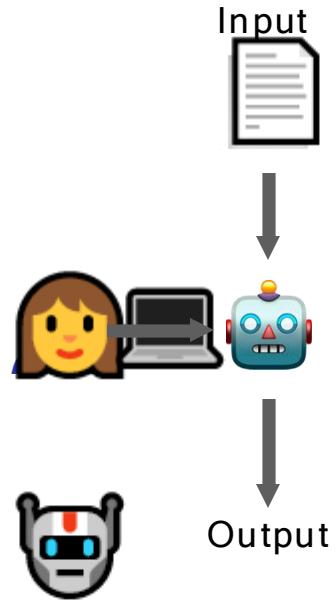


---

# **Large Language Models**

**CPEN - 320**

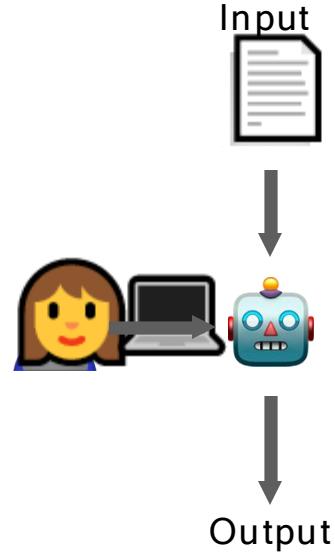
# Traditional Programming



2

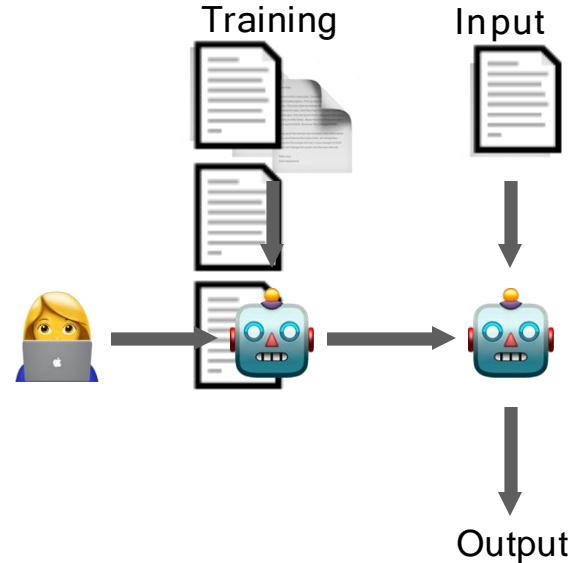
Software 1.0

# Traditional Programming vs Machine Learning



Software 1.0

3



Software 2.0

# Types of Machine Learning

## Unsupervised Learning

Learn structure of data to predict (e.g., clustering)

## Supervised Learning

Learn how data maps to labels to recognize or predict

## Reinforcement Learning

Learn how to act in an environment to obtain reward

4

"This product does what it is supposed   "



→ *cat*



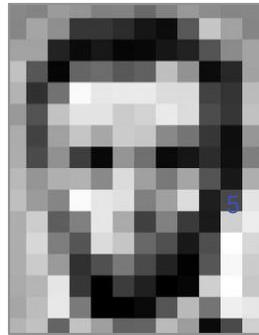
→ "Hey  
*Siri*"



# Embedding: Inputs and outputs are always numbers

**What we see**

**Input**

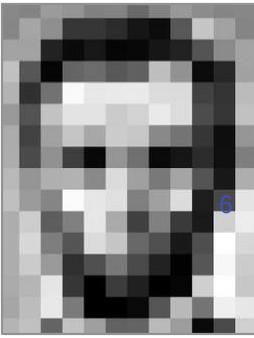


**Output**

"Lincoln"

# Embedding: Inputs and outputs are always numbers

What we see



Input

Output

"Lincoln"

What the machine "sees"

157	153	174	168	150	152	129	151	172	161	166
155	182	163	74	75	62	33	17	110	210	180
180	180	50	14	34	6	10	33	48	106	159
206	109	5	124	131	111	120	204	166	15	56
194	68	137	251	237	239	239	228	227	87	71
172	105	207	239	239	214	220	239	228	98	74
188	88	179	209	185	215	211	158	139	75	20
189	97	165	84	10	168	134	11	31	62	22
199	168	191	193	158	227	178	143	182	106	36
205	174	155	252	236	231	149	178	228	43	95
190	216	116	149	236	187	86	150	79	38	218
190	224	147	108	227	210	127	102	36	101	255

[76, 105, 110, 99, 111, 108, 110]

# Why is this hard?

- Infinite variety of inputs can all mean the same thing
- Meaningful differences can be tiny
- Structure of the world is complex



*"I loved this movie"*

*"As good as The Godfather"*



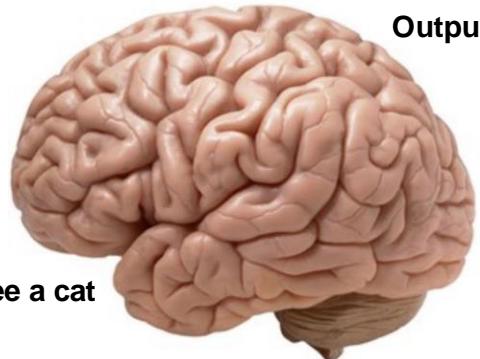
@teenybiscuit

# How is it done?

- Many methods for Machine Learning
  - Logistic Regression
  - Support Vector Machines<sup>8</sup>
  - Decision Trees
- But one is dominant
  - Neural Networks (also called Deep Learning)

# Inspiration

• Inspired by



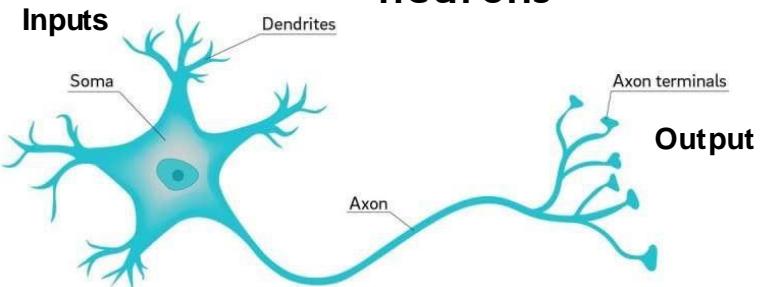
Input: see a cat

Output: say "cat"

S  
of  
9

<https://www.the-scientist.com/the-nutshell/what-makes-human-brains-so-big-36663>

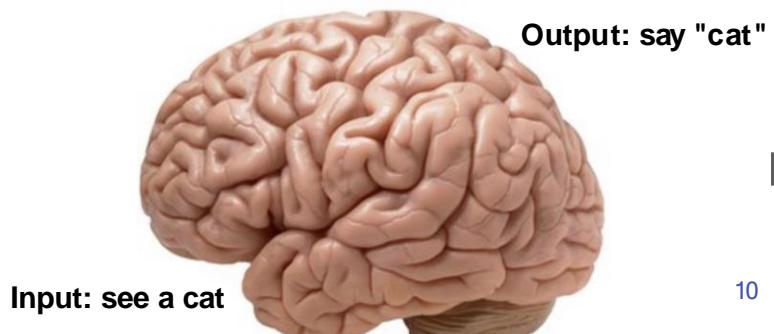
billions of  
neurons



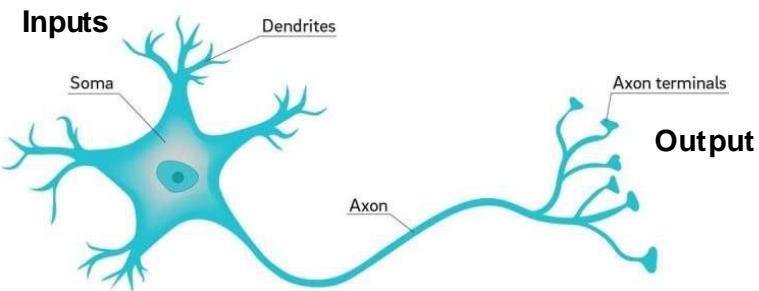
<https://medicalxpress.com/news/2018-07-neuro-scientists-optimize-brain.html>

sends an  
electrical

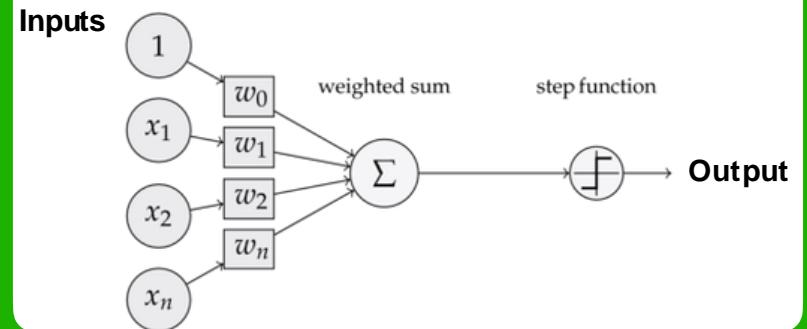
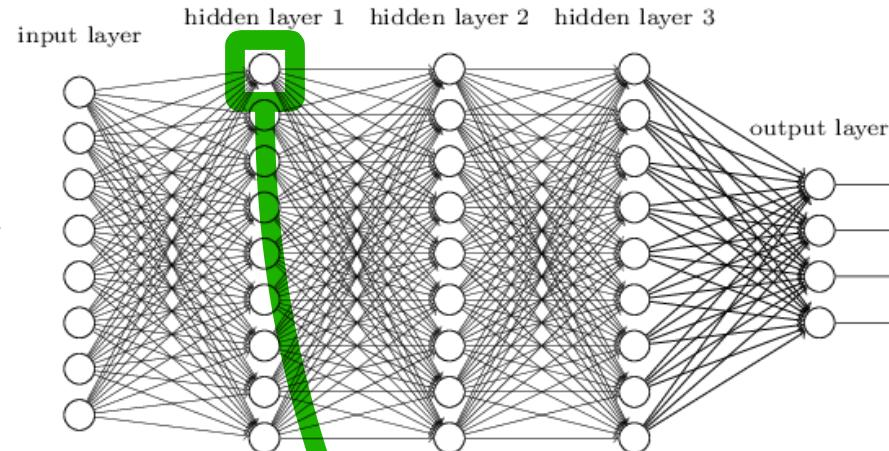
# Formalization



<https://www.the-scientist.com/the-nutshell/what-made-human-brains-so-big-36663>

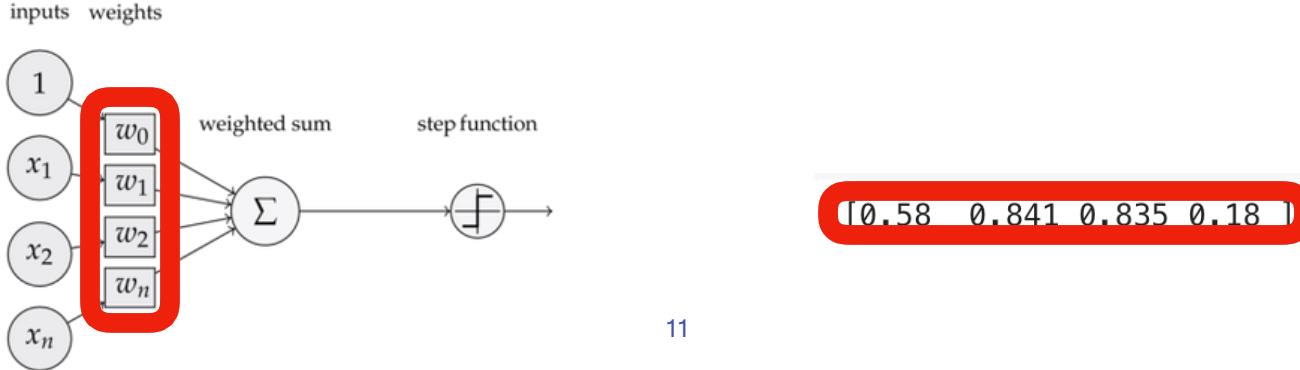


<https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html>



<https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/>

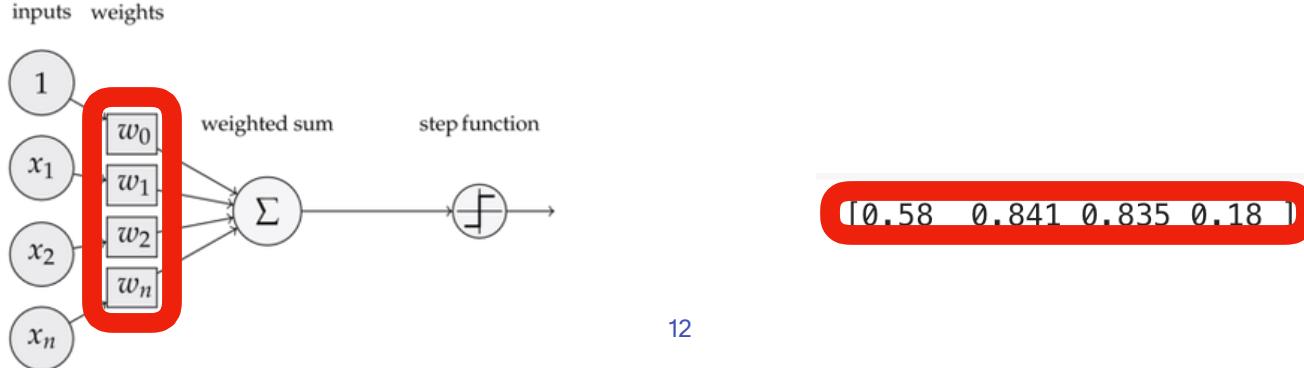
# A "perceptron" is a vector of numbers



11

<https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/>

# Step Function



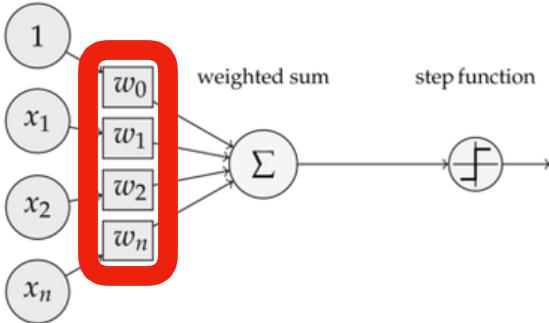
12

<https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/>

- Step function: also called threshold function
  - Decides based on the weighted sum if the output is 0 or 1
- In modern deep learning, the step function is less often used as it is not differentiable at the threshold point
- Instead, smoother functions that approximate the step function but retain differentiability are used
- For example: **sigmoid** outputs a continuous value in the **range** (0, 1)

# A "layer" is a matrix of numbers

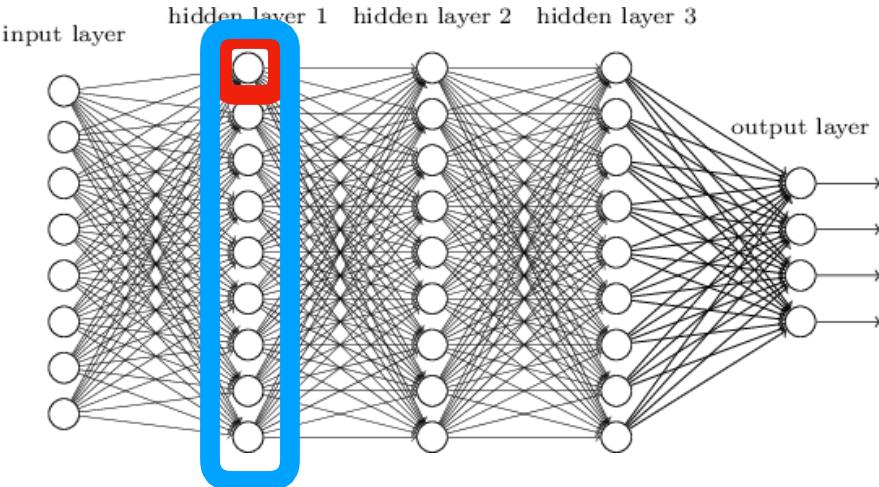
inputs weights



weighted sum

13

<https://www.jessicayung.com/explaining-tensorflow-code-for-a-multilayer-perceptron/>



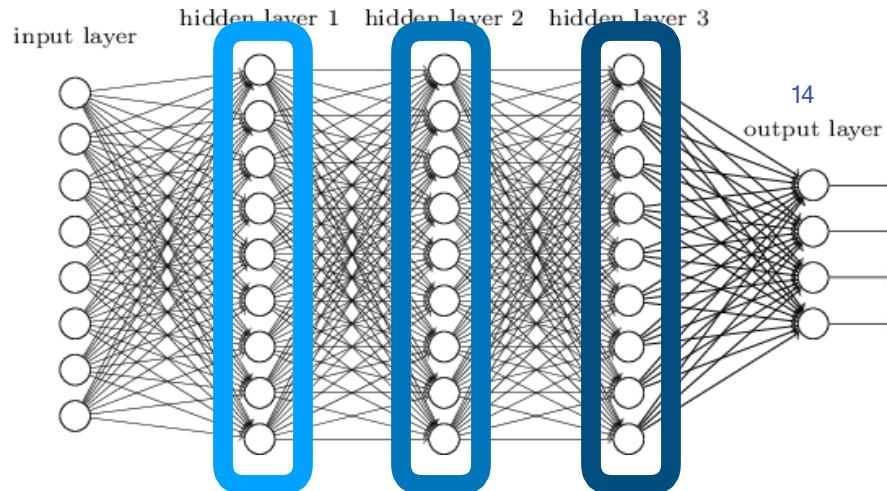
input layer    hidden layer 1    hidden layer 2    hidden layer 3

output layer

[0.58 0.841 0.835 0.18]
[0.405 0.813 0.309 0.562]
[0.422 0.229 0.46 0.152]
[0.673 0.429 0.441 0.243]
[0.9 0.744 0.234 0.856]
[0.971 0.486 0.175 0.248]
[0.258 0.588 0.478 0.266]
[0.236 0.496 0.077 0.557]
[0.413 0.322 0.372 0.741]

# The neural network is a set of matrices

Called "parameters" or "weights"

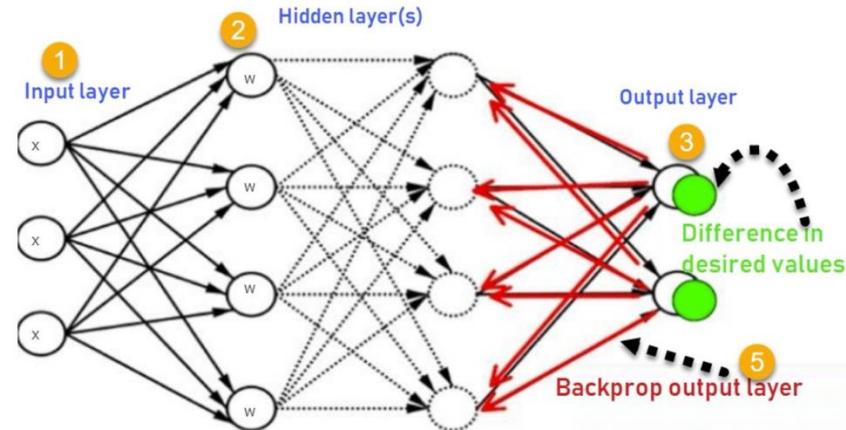


[[0.32 0.866 0.151 0.555]
[ [0.58 0.841 0.825 0.18 1]
[ [0.32 0.866 0.151 0.555]
[ [0.467 0.414 0.562 0.555]
[ [0.186 0.937 0.131 0.448]
[ [0.892 0.567 0.111 0.678]
[ [0.483 0.478 0.424 0.844]
[ [0.774 0.963 0.213 0.569]
[ [0.014 0.509 0.21 0.26 ]
[ [0.142 0.991 0.105 0.211]
[ [0.789 0.242 0.774 0.019] ]

NN operations are just matrix multiplications.  
GPUs are really fast at matrix multiplications.

# Training

- **Dataset** contains n items
- **Batch**: a subset of n to use for training
  - If n is large, it won't fit in memory
- **Epoch**: going over the entire n items in the dataset

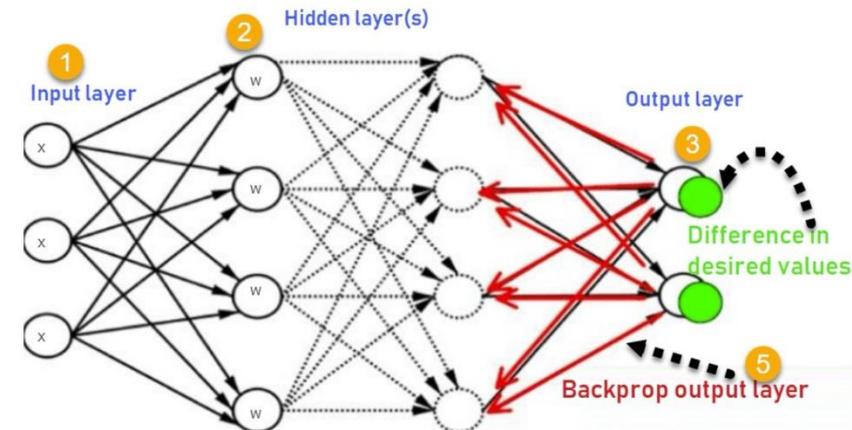
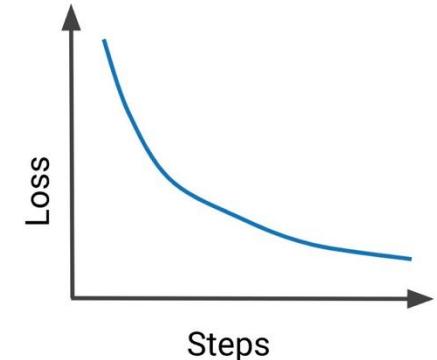


<https://www.guru99.com/backpropogation-neural-network.html>

# Training

$$loss_{CE} = - \sum (y_i * \log(y'_i))$$

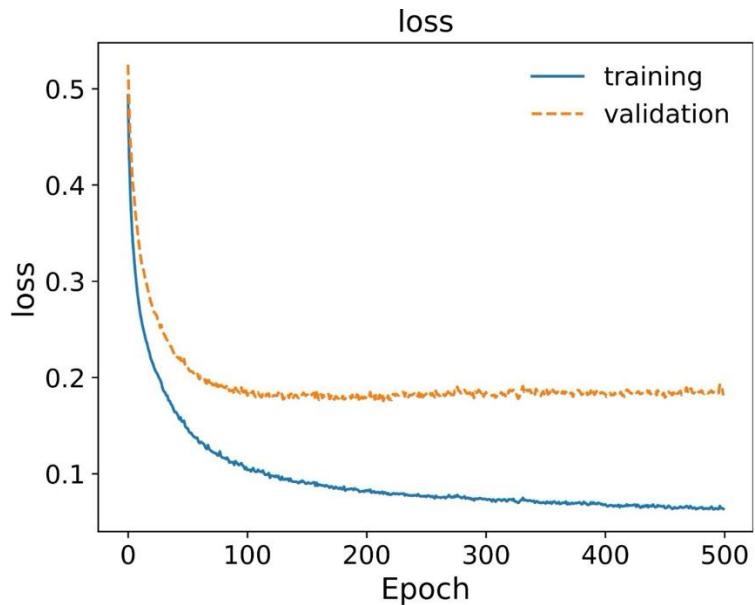
- Data X (e.g. images), labels y (e.g. labels)
- Take a little batch of data x:
  - Use the current model to make a prediction  $x \rightarrow y'$
  - Compute  $loss(y, y')$
  - Back-propagate the loss through all the layers of the model
- Repeat until loss stops decreasing



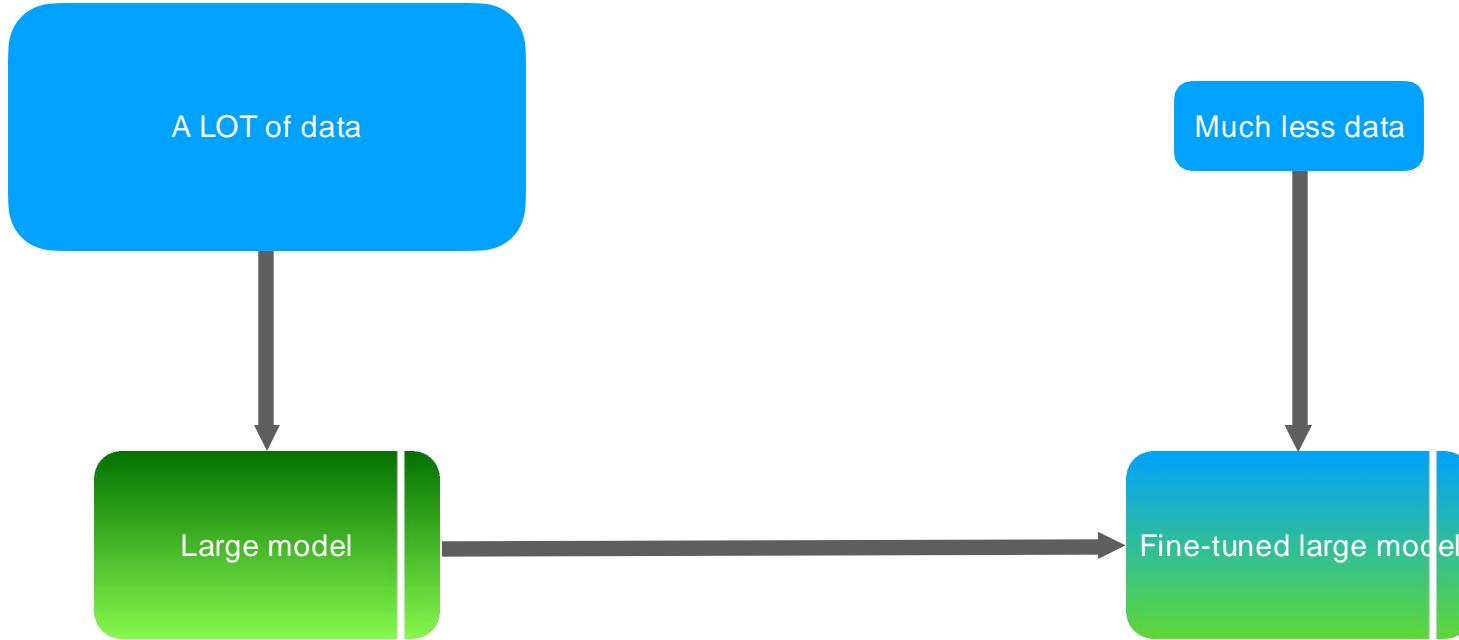
<https://www.guru99.com/backpropagation-neural-network.html>

# Dataset Splitting

- Split  $(X, y)$  into training ( $\sim 80\%$ ), validation ( $\sim 10\%$ ), and test ( $\sim 10\%$ ) sets
- Validation set is for
  - ensuring that training is not "overfitting"
  - setting hyper-parameters of the model (e.g. number of parameters)
- Test set is for measuring validity of predictions on new data



THIS APPLIES TO YOUR  
EXPERIMENTATION  
WITH PROMPTS!



**Pre-training:**  
slow training on a lot of data

**Fine-tuning:**  
fast training on a little data

# Model Hubs

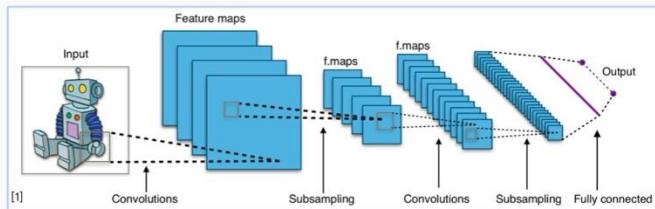
- People share pre-trained models!
  - 180K models
  - 30K datasets

The screenshot shows the Hugging Face Model Hub interface. At the top, there is a search bar with the placeholder "Search models, datasets, users...". Below the search bar are navigation tabs: "Models" (selected), "Datasets", "Spaces", "Docs", "Solutions", and "Principles". On the left side, there is a sidebar with a "Tasks" tab (selected) and other tabs: "Libraries", "Datasets", "Languages", "Licenses", and "Other". Under the "Tasks" tab, there are sections for "Multimodal" tasks like Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Document Question Answering, and Graph Machine Learning; "Computer Vision" tasks like Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, and Zero-Shot Image Classification; and "Natural Language Processing" tasks like Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation, Summarization, Conversational, Text Generation, Text2Text Generation, Fill-Mask, Sentence Similarity, Text-to-Speech, and Automatic Speech Recognition. On the right side, the main content area displays a list of 20,064 models. Each model entry includes the owner's profile picture, the model name, a short description, the last update date, the number of downloads, and the number of stars. Some entries also show the number of commits. The models listed include "cardiffnlp/twitter-roberta-base-sentiment", "Seethal/sentiment\_analysis\_generic\_dataset", "cardiffnlp/twitter-roberta-base-sentiment-latest", "yiyanghkust/finbert-tone", "ProsusAI/finbert", "zhayunduo/roberta-base-stocktwits-finetuned", "nlptown/bert-base-multilingual-uncased-sentiment", "prithivida/parrot\_adequacy", "roberta-base-openai-detector", "cross-encoder/ms-marco-MiniLM-L-12-v2", and "oliverguhr/german-sentiment-bert".

# Before ~2020: each task had its own NN architecture

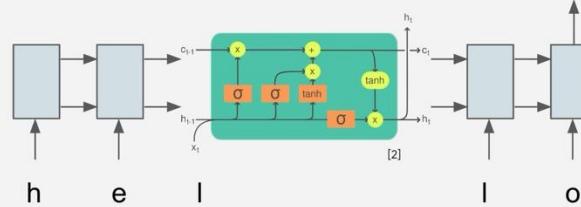
## Computer Vision

Convolutional NNs (+ResNets)



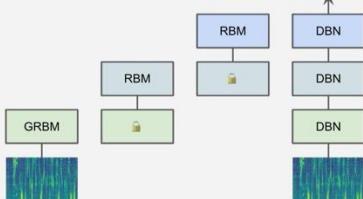
## Natural Lang. Proc.

Recurrent NNs (+LSTMs)



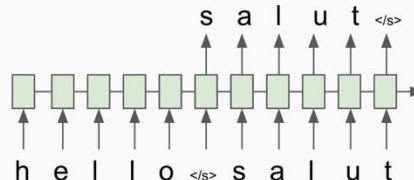
## Speech

Deep Belief Nets (+non-DL)



## Translation

Seq2Seq



## RL

BC/GAIL

**Algorithm 1** Generative adversarial imitation learning

- 1: **Input:** Expert trajectories  $\tau_E \sim \pi_E$ , initial policy and discriminator parameters  $\theta_0, w_0$
  - 2: **for**  $t = 0, 1, 2, \dots$  **do**
  - 3:   Sample trajectories  $\tau_t \sim \pi_\theta$
  - 4:   Update the discriminator parameters from  $w_t$  to  $w_{t+1}$  with the gradient
- $$\hat{\mathbb{E}}_{\tau_t} [\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E} [\nabla_w \log(1 - D_w(s, a))] \quad (17)$$
- 5:   Take a policy step from  $\theta_t$  to  $\theta_{t+1}$ , using the TRPO rule with cost function  $\log(D_{w_{t+1}}(s, a))$ . Specifically, take a KL-constrained natural gradient step with
$$\hat{\mathbb{E}}_{\tau_t} [\nabla_\theta \log \pi_\theta(a|s) Q(s, a)] - \lambda \nabla_\theta H(\pi_\theta), \quad (18)$$

where  $Q(s, a) = \hat{\mathbb{E}}_{\tau_t} [\log(D_{w_{t+1}}(s, a)) | s_0 = s, a_0 = a]$
- 
- 6: **end for**

# Now: all is Transformers



Transformer cartoon (DALL-E)

# The Transformer Architecture



# Attention is all you need (2017)

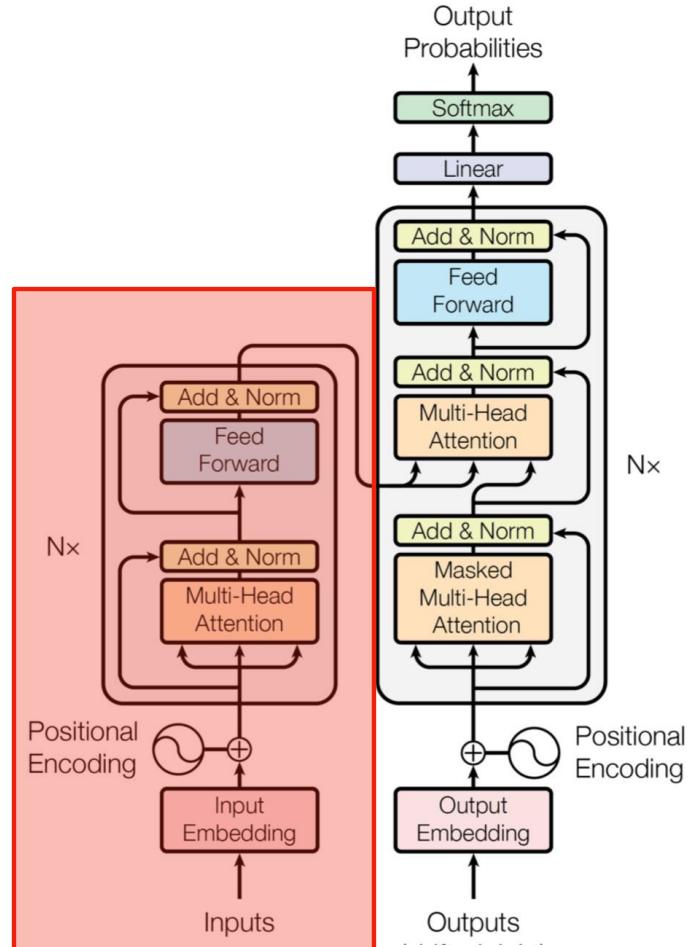
<https://arxiv.org/abs/1706.03762>

- How does the Transformer work?

# Attention is all you need (2017)

<https://arxiv.org/abs/1706.03762>

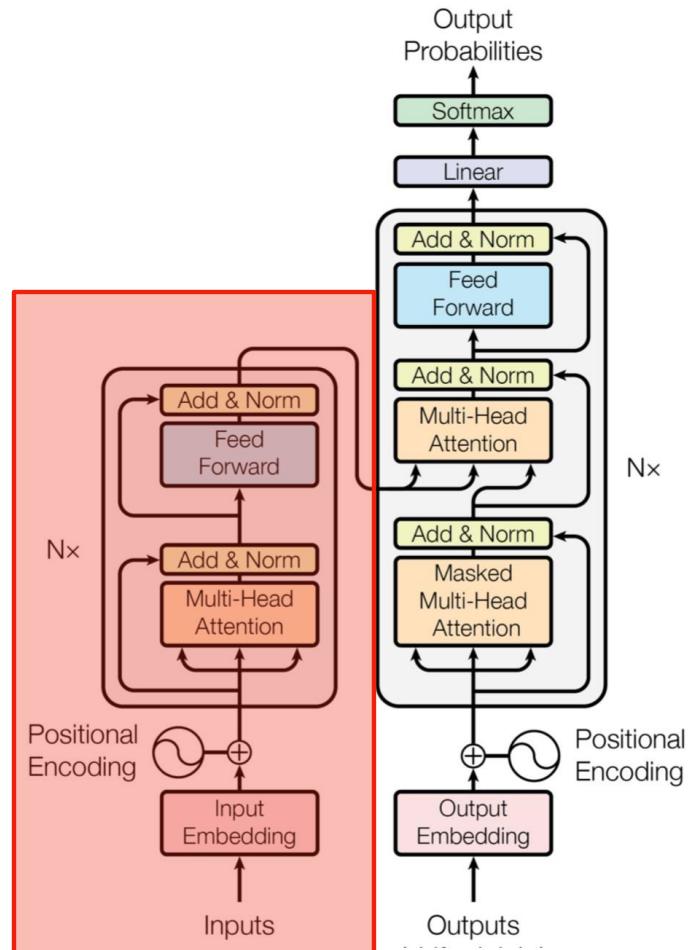
- Ground-breaking architecture that set SOTA on first translation and later all other NLP tasks



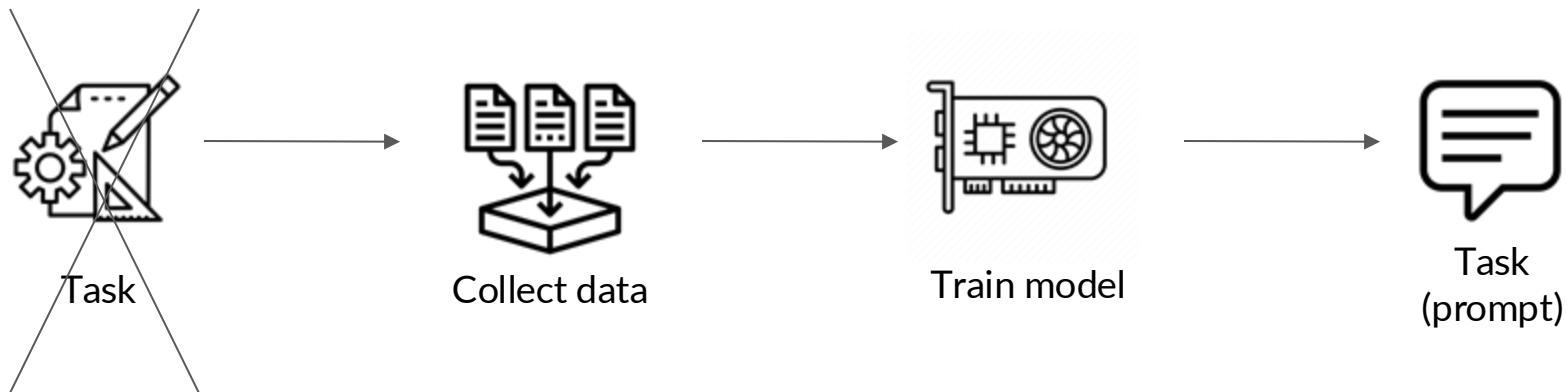
# Transformer

A breakthrough in AI that changes how we process data.

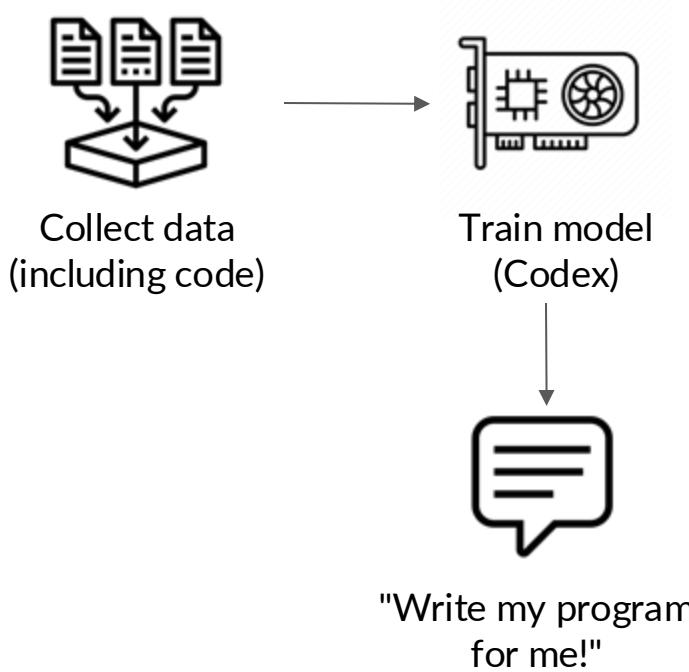
The core idea: Instead of reading word by word, transformers focus on the whole sentence and **important** parts of a sentence all at once.



# Foundation Models - A New Paradigm



# Foundation Models



Produce human-like text



Speech to text translation

**Whisper**

Creates images from text

**DALL-E**

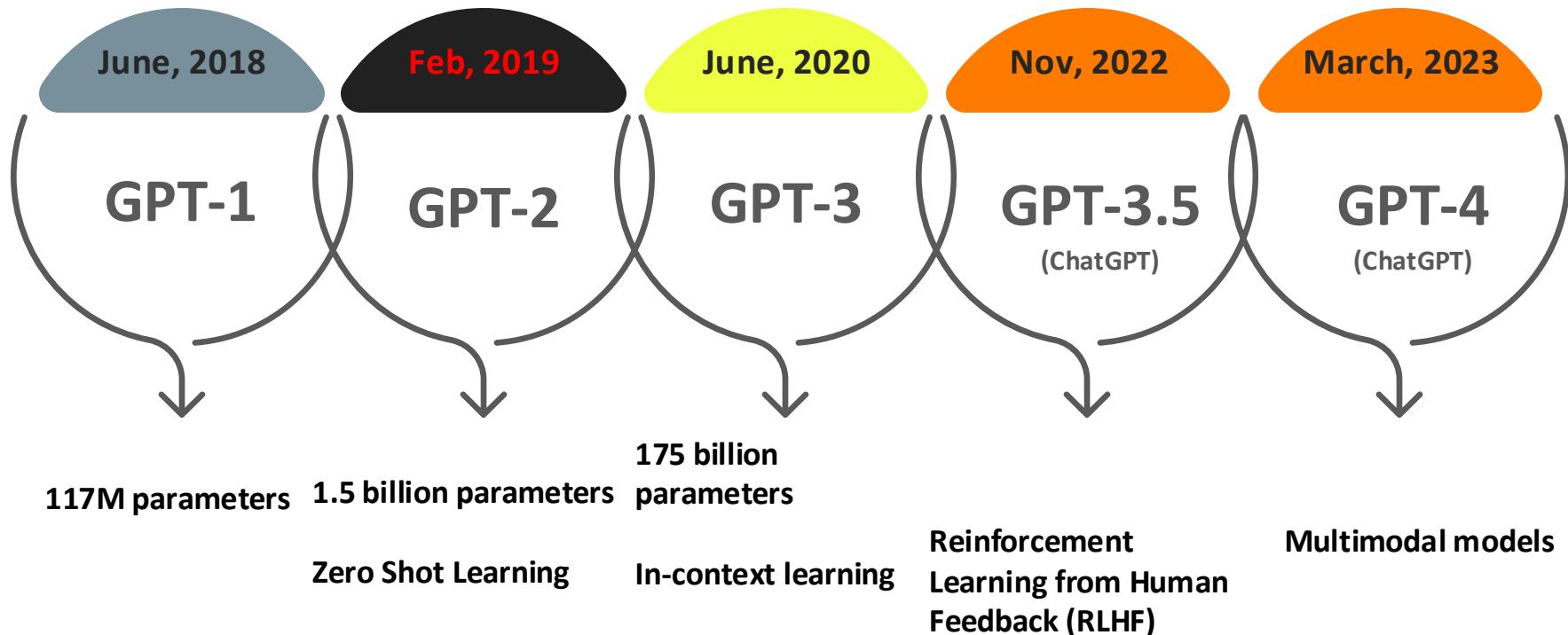
Create description for an image

**CLIP**

Write code from text description

**Codex**

# GPT Timeline



# Interacting with LLMs



# Prompting

- Model fine-tuning used to be necessary
- Larger (or instruction tuned) models give intelligible responses even without it - just prompt them!

The three settings we explore for in-context learning

## 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.



Traditional fine-tuning (not used for GPT-3)

## Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Taken from the GPT-3 paper:  
<https://arxiv.org/pdf/2005.14165.pdf>

# Few-shot Prompting?

# Few-shot Prompting

Zero-Shot

[Instruction] [Input]

*Is the sentiment positive or negative?*

*“This movie sucks!” A:*

---

One-Shot No-Instruction

[Ex In 1] [Ex Out 1] [Input]

Q: *This movie rocks!* A: *Positive.*

Q: *“This movie sucks!”* A:

---

One-Shot

[Instruction] [Ex In 1] [Ex Out 1]  
[Input]

*Is the sentiment positive or negative?* Q: *This movie rocks!* A: *Positive.* Q: *“This movie sucks!”*

A:

---

Two-Shot

[Instruction] [Ex In 1] [Ex Out  
1] [Ex In 2] [Ex Out 2] [Input]

*Is the sentiment positive or negative?* Q: *This movie rocks!* A: *Positive.* Q: *My eyes are bleeding!* A: *Negative.* Q: *“This movie sucks!”* A:

---

Chain-of-Thought

[Instruction]  
[Request step-by-step  
explanation] [Input]

*Is the sentiment positive or negative? Please explain your answer step-by-step.* Q: *“This movie sucks!”* A:

# Prompting examples



# Prompt Example 1 – Grammar Correction

Q: Correct this to standard English:

*Anna and Mike is going skiing.*

**Model:** Anna and Mike are going skiing



## Prompt Example 2 – Summarization

Large language models (LLMs) have shown promise for automatic summarization but the reasons behind their successes are poorly understood. By conducting a human evaluation on ten LLMs across different pretraining methods, prompts, and model scales, we make two important observations. First, we find instruction tuning, and not model size, is the key to the LLM's zero-shot summarization capability. Second, existing studies have been limited by low-quality references, leading to underestimates of human performance and lower few-shot and finetuning performance. To better evaluate LLMs, we perform human evaluation over high-quality summaries we collect from freelance writers. Despite major stylistic differences such as the amount of paraphrasing, we find that LMM summaries are judged to be on par with human written summaries.

What is the main idea of the article?

# Prompt Example 3 – Style Transfer

Here is some text: {When Mohammed left the theatre, it was already dark out.}. Rewrite it to be more about the movie itself.

{The movie was longer than Mohammed had expected, and despite the excellent ratings he was a bit disappointed when he left the theatre.}.



Here is some text: {next to the path}. Rewrite it to be about France.

{next to la Siene}.

Here is some text: {The man stood outside the grocery store, ringing the bell.}. Rewrite it to be about clowns.

{The man stood outside the circus, holding a bunch of balloons.}.

Here is some text: {the bell ringing}. Rewrite it to be more flowery.

{the peales of the jangling bell}.

Here is some text: {against the tree}. Rewrite it to be include the word "snow".

{against the snow-covered bark of the tree}.

Here is some text: {We should increase the government tax rate on profits earned from the sale of stocks, bonds, and real estate.}. Rewrite it in the style of Kanye West.

# Prompt Example 4 – “Let’s Think Step By Step.”

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 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.

There are 16 balls total.

Half of them (8) are golf balls.

Half of those (4) are blue.

Therefore, there are 4 blue golf balls.

# Prompt Example 5 – Joke Explanation

## Explaining a Joke

**Input:** Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!

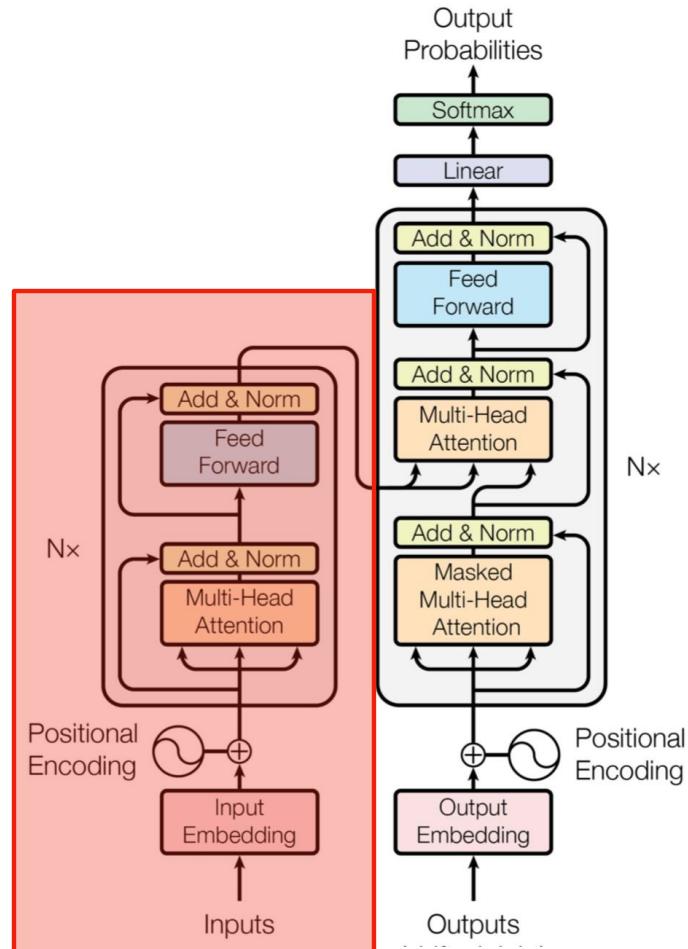
**Model Output:** TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

# HERE NOV 2024

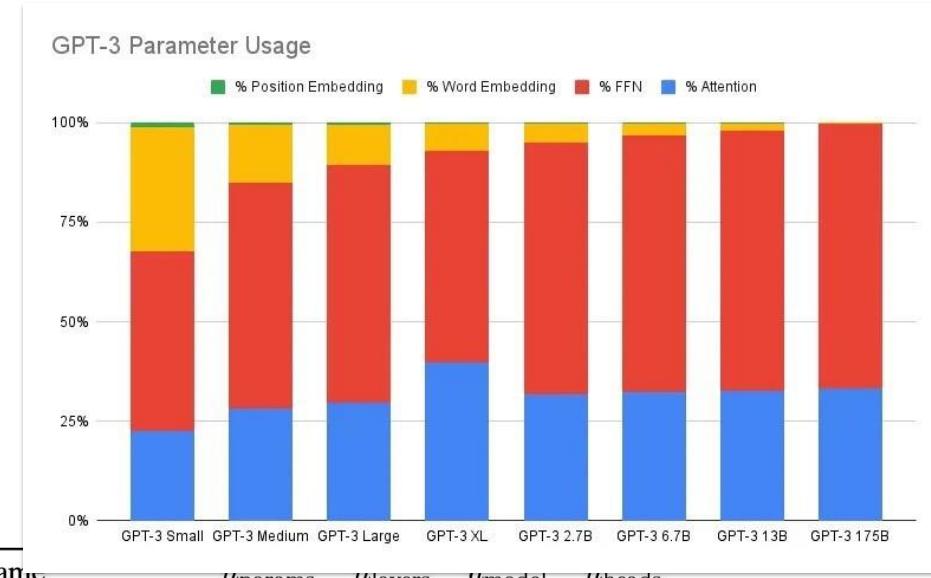
# Transformer architecture

A breakthrough in AI that changes how we process data.

The core idea: Instead of reading word by word, transformers focus on the whole sentence and **important** parts of a sentence all at once.



# Transformer Architecture



Model Name	#params	#layers	#model	#heads
GPT-3 Small	125M	12	768	12
GPT-3 Medium	350M	24	1024	16
GPT-3 Large	760M	24	1536	16
GPT-3 XL	1.3B	24	2048	24
GPT-3 2.7B	2.7B	32	2560	32
GPT-3 6.7B	6.7B	32	4096	32
GPT-3 13B	13.0B	40	5140	40
GPT-3 175B or "GPT-3"	175.0B	96	12288	96



# Why does this work so well?



Andrej Karpathy ✅

@karpathy

...

The Transformer is a magnificent neural network architecture because it is a general-purpose differentiable computer. It is simultaneously:

- 1) expressive (in the forward pass)
- 2) optimizable (via backpropagation+gradient descent)
- 3) efficient (high parallelism compute graph)

11:54 AM · Oct 19, 2022

---

490 Retweets

39 Quotes

3,670 Likes

1,023 Bookmarks

# We mostly don't understand it, though

## In-context Learning and Induction Heads

### AUTHORS

Catherine Olsson\*, Nelson Elhage\*, Neel Nanda\*, Nicholas Joseph†, Nova DasSarma†, Tom Henighan†, Ben Mann†, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, Chris Olah\*

### AFFILIATION

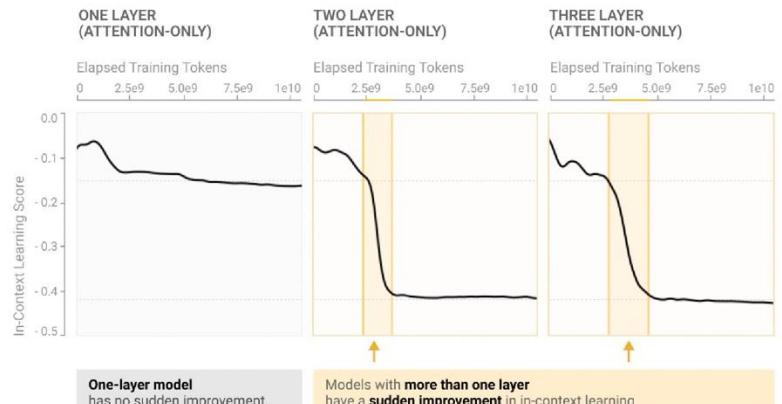
Anthropic

### PUBLISHED

Mar 8, 2022

\* Core Research Contributor; † Core Infrastructure Contributor; \* Correspondence to colah@anthropic.com;  
Author contributions statement below.

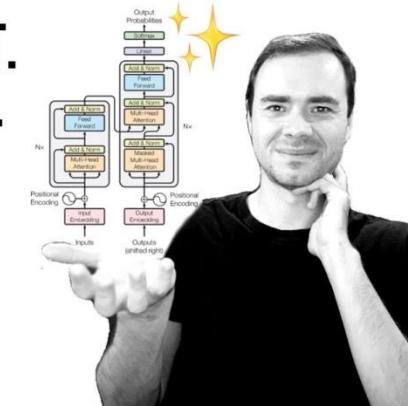
### MODELS WITH MORE THAN ONE LAYER HAVE AN ABRUPT IMPROVEMENT IN IN-CONTEXT LEARNING



# Should you be able to code a Transformer?

- Definitely not necessary!
- BUT: it's not difficult, it is fun, and is probably worth doing
- Andrej Karpathy's GPT-2 implementation is <400 lines of code, including Attention

**LET'S BUILD GPT.  
FROM SCRATCH.  
IN CODE.  
SPELLED OUT.**



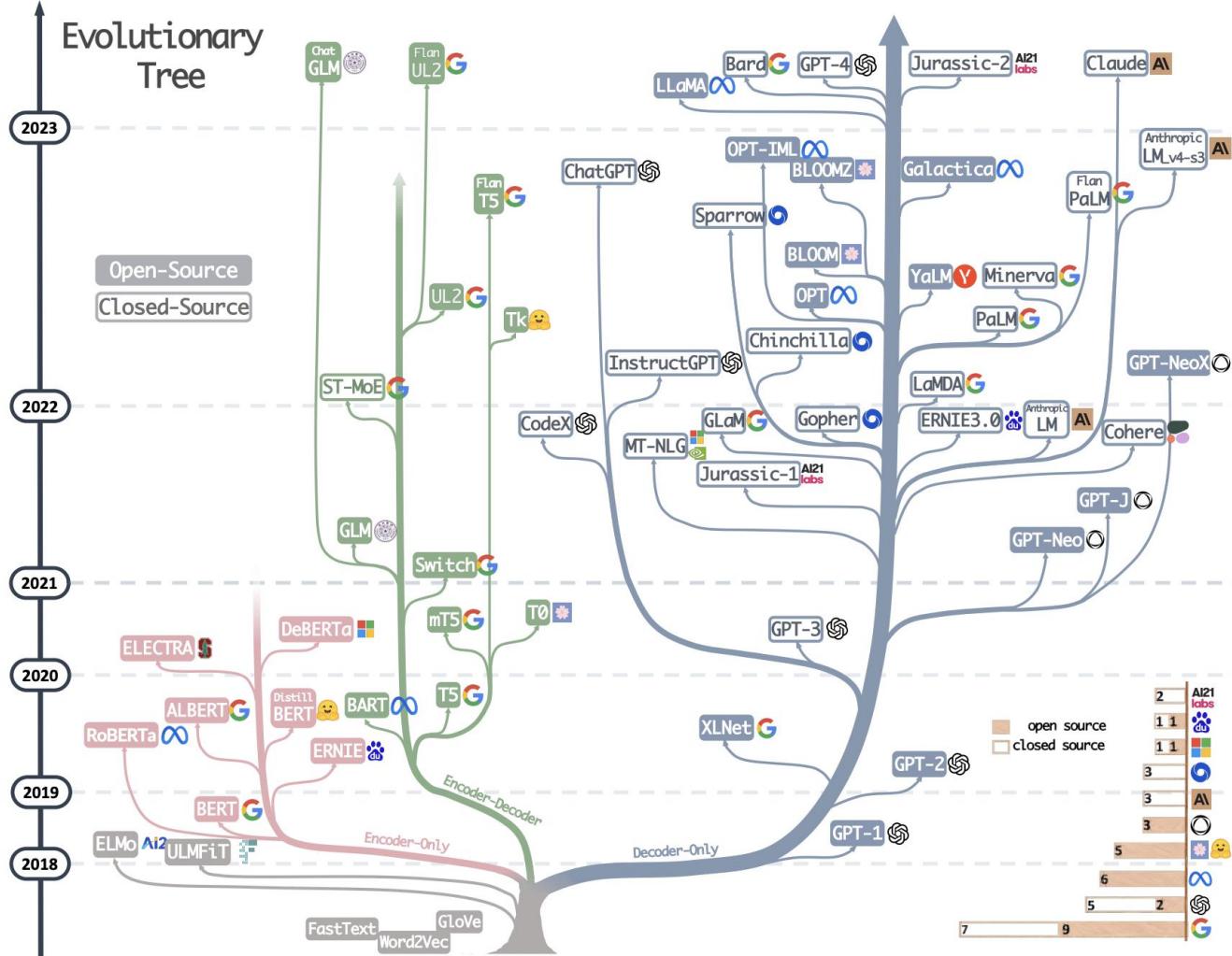
<https://www.youtube.com/watch?v=kCc8FmEb1nY>

# Notable Large Language Models (LLMs)

# What does GPT stand for?

Generative Pre-Trained  
Transformer (GPT)

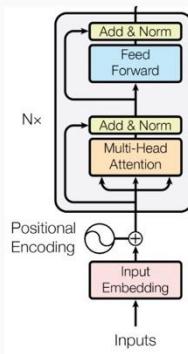




# Three architectures

## BERT

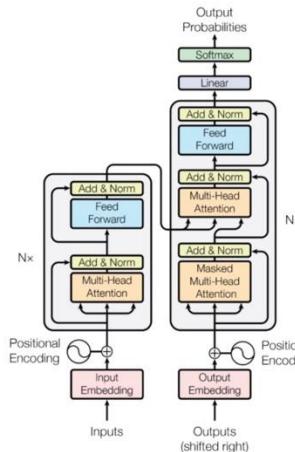
[\*] [\*] [sat\_] [\*] [the\_] [\*]



[The\_] [cat\_] [MASK] [on\_] [MASK] [mat\_]

## T5

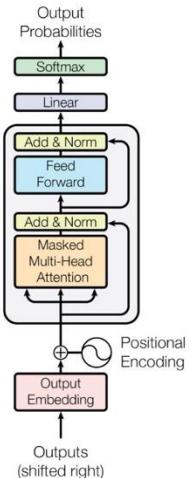
Das ist gut.  
A storm in Attala caused 6 victims.  
This is not toxic.



Translate EN-DE: This is good.  
Summarize: state authorities dispatched...  
Is this toxic: You look beautiful today!

## GPT

[sat\_]

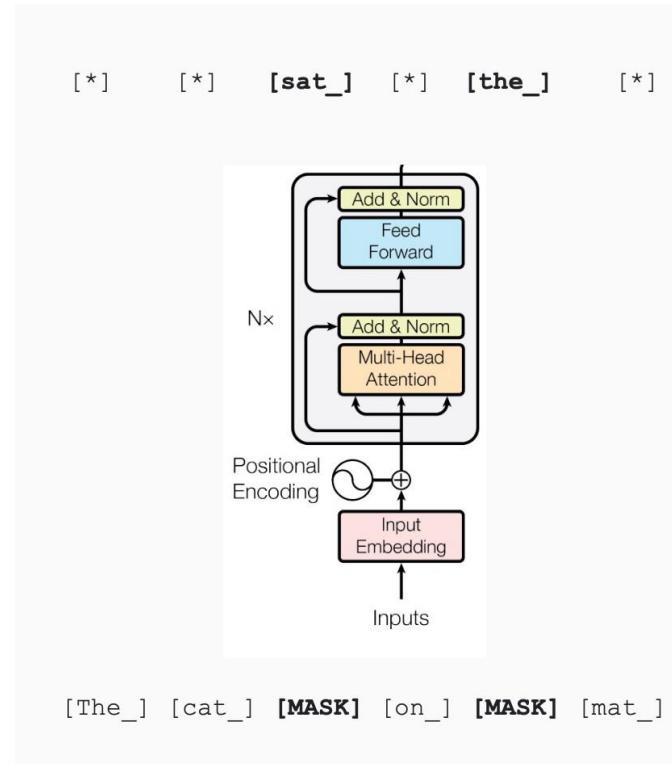


[START] [The\_] [cat\_]

Transformer image source: "Attention Is All You Need" paper

# BERT (2019)

- Encoder-only
- Focuses on processing and understanding the input sequence.
- 110M params
- Text classification (e.g., sentiment analysis)
- Was great, now dated



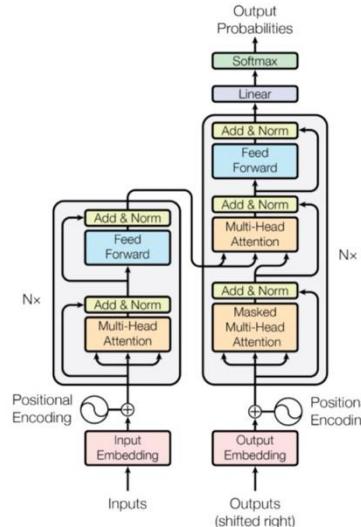
# T5: Text-to-Text Transfer Transformer (2020)

- Google's original Transformer architecture
- Encoder-Decoder architecture
- Input and output are both text strings
- 11B parameters
- Machine translation, summarization
- Still could be a good choice for fine-tuning!

Das ist gut.

A storm in Attala caused 6 victims.

This is not toxic.



Translate EN-DE: This is good.

Summarize: state authorities dispatched...

Is this toxic: You look beautiful today!

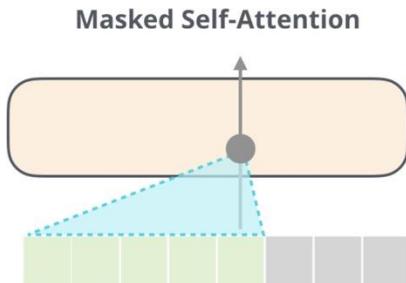
# GPT / GPT-2 (2019)

[sat\_]

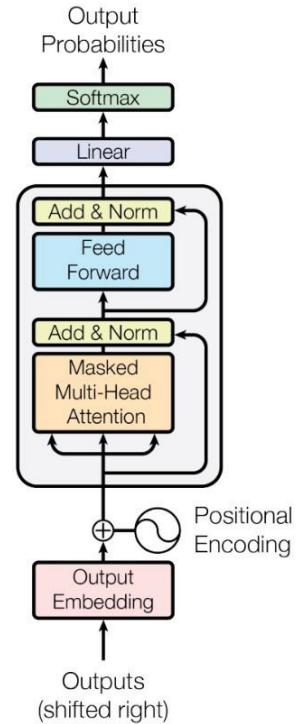
- Generative Pre-trained Transformer
- Decoder-only!
- 1.5B

Specifically, for a given token in the sequence, it can only attend to:

- Itself and
- The tokens that came before it (not the tokens that come after it).

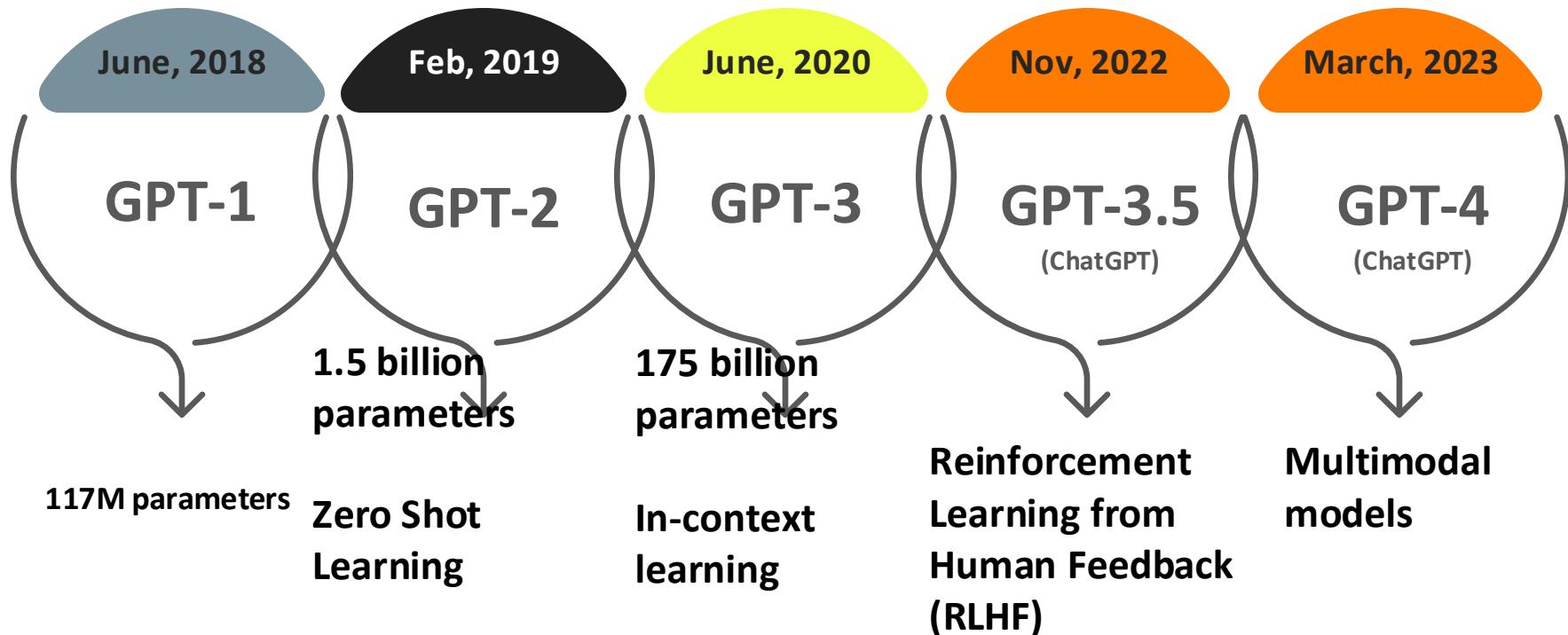


[START] [The\_] [cat\_]



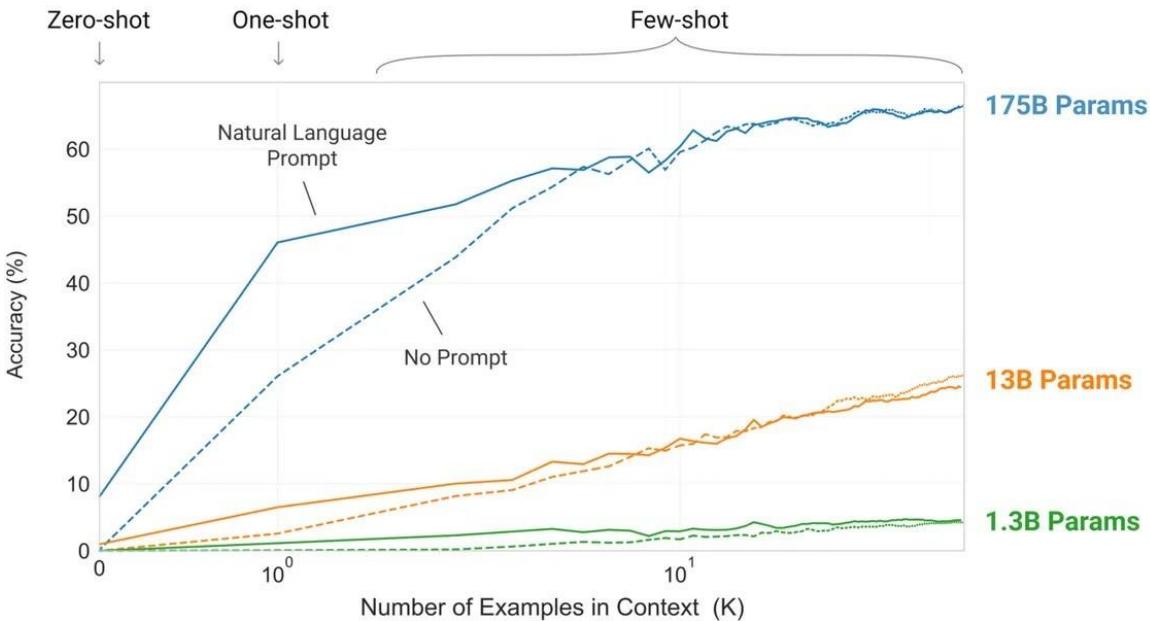
From Lukas Beyer's lecture *Intro to Transformers*.

# GPT Timeline



# GPT-3 (2020)

- Just like GPT-2, but 100x larger (175B params)
- Exhibited unprecedented few-shot and zero-shot learning



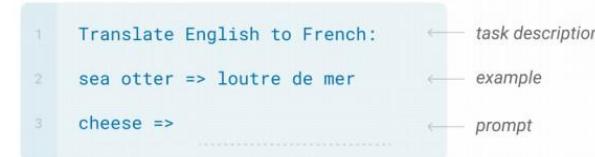
## Zero-shot

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



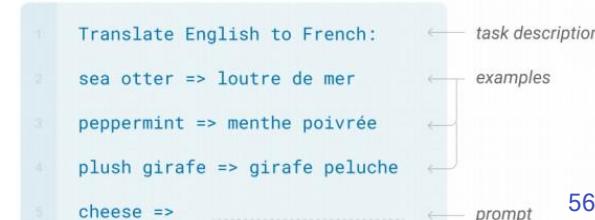
## One-shot

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.



# GPT-3 Training Data

- For a total of 500B tokens
- But trained on only 300B!

Wikipedia (facts) (3.49%)

Books1/BookCorpus (Smashwords) (7.8%)

Books2 (Libgen or similar) (8.1%)

WebText (Reddit links) (18.86%)

Common Crawl (www) (61.75%)

# GPT-4 (2023)

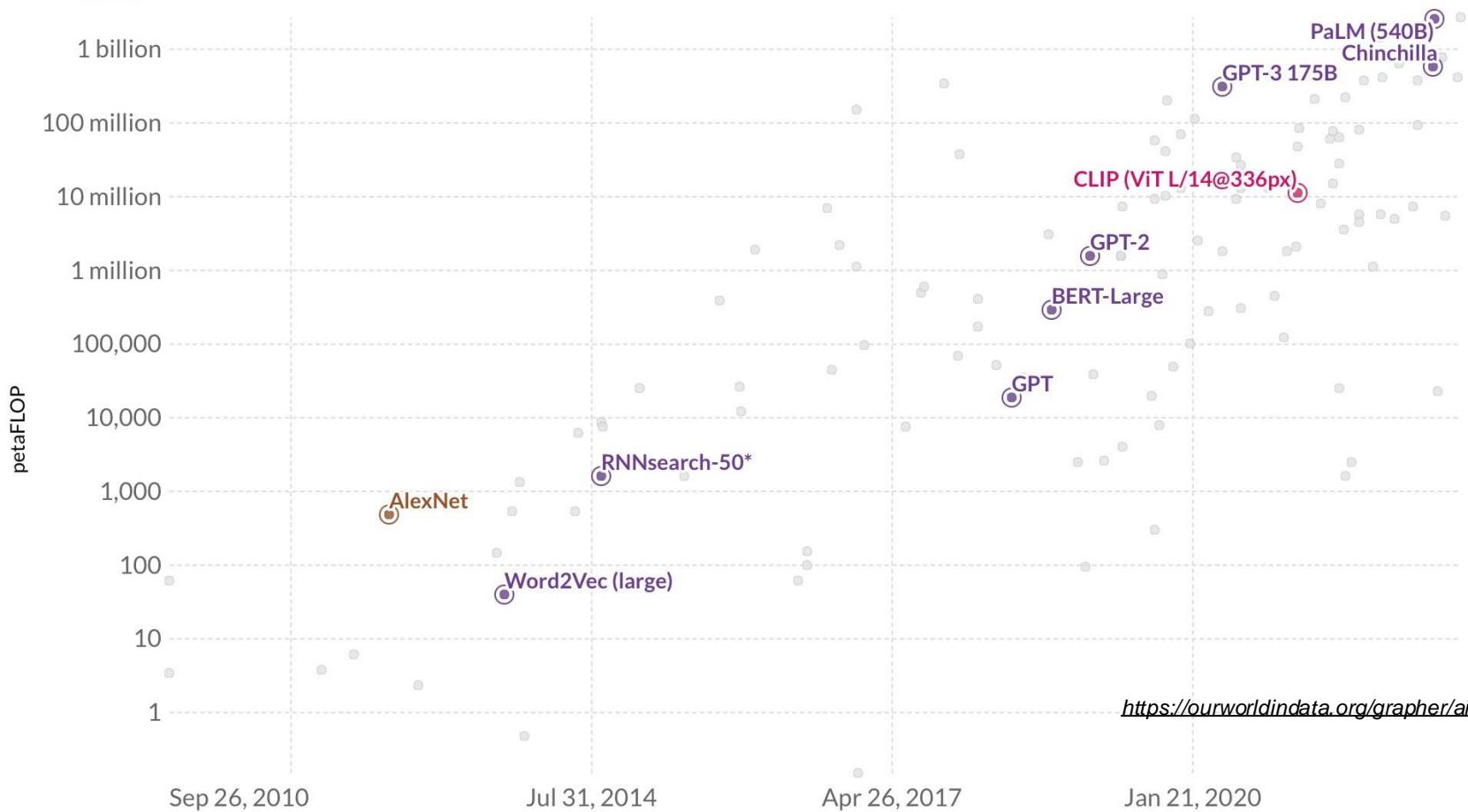
This report focuses on the capabilities, limitations, and safety properties of GPT-4. GPT-4 is a Transformer-style model [39] pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF) [40]. Given both the competitive landscape and the safety implications of large-scale models like GPT-4, this report contains no further details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar.



# Computation used to train notable AI systems

Computation is measured in petaFLOP, which is  $10^{15}$  floating-point operations.

LINEAR  LOG  Select systems  Zoom to selection



<https://ourworldindata.org/grapher/ai-training-computation>

But what exactly is the  
relationship  
between  
model size  
and dataset  
size?

But what exactly is the relationship between model size and dataset size?

Larger models tend to perform better, but the gains from increasing model size diminish unless the dataset size also increases.

# LLaMA Training Data

- Custom quality-filtering of CommonCrawl + some C4 + Github + Wikipedia + Books + ArXiV + Stack Exchange
- RedPajama: open-source recreation

	RedPajama	LLaMA*
CommonCrawl	878 billion	852 billion
C4	175 billion	190 billion
Github	59 billion	100 billion
Books	26 billion	25 billion
ArXiv	28 billion	33 billion
Wikipedia	24 billion	25 billion
StackExchange	20 billion	27 billion
Total	1.2 trillion	1.25 trillion



Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

*hĒĒps://arxiv.org/pdf/2302.13971.pdf*

# Including code in training data

- T5 and GPT-3 (2020) specifically removed code. But most recent models are trained on ~5% code. Why?
  - Code-specific models such as OpenAI Codex (2021): GPT-3 further trained on public GitHub code.
  - Empirically, this improved performance on non-code tasks!
  - Open-source dataset: The Stack (3TB of permissively licensed source code)

# BigCode @BigCodeProject

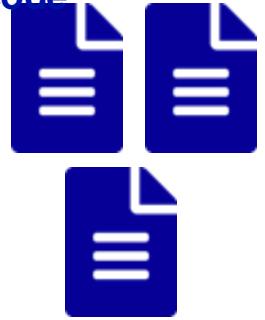
# GPT Models: How Are They Trained?

How is GPT Created?

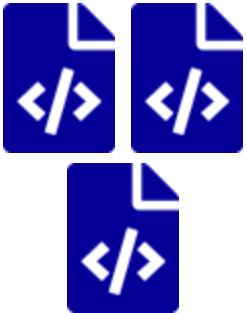


# Transition to ChatGPT

Lots of web text  
code



Lots of GitHub



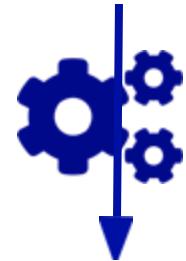
Lots of annotated  
data



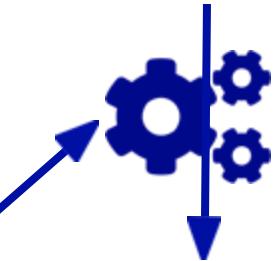
Human judgements  
of response quality



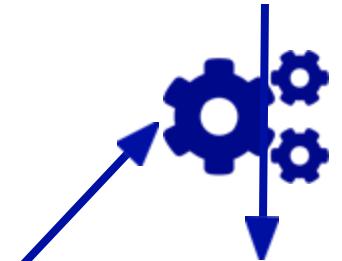
Chat-oriented data



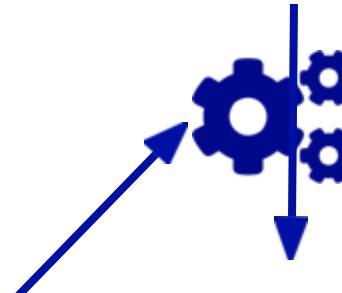
davinci



code-  
davinci-002



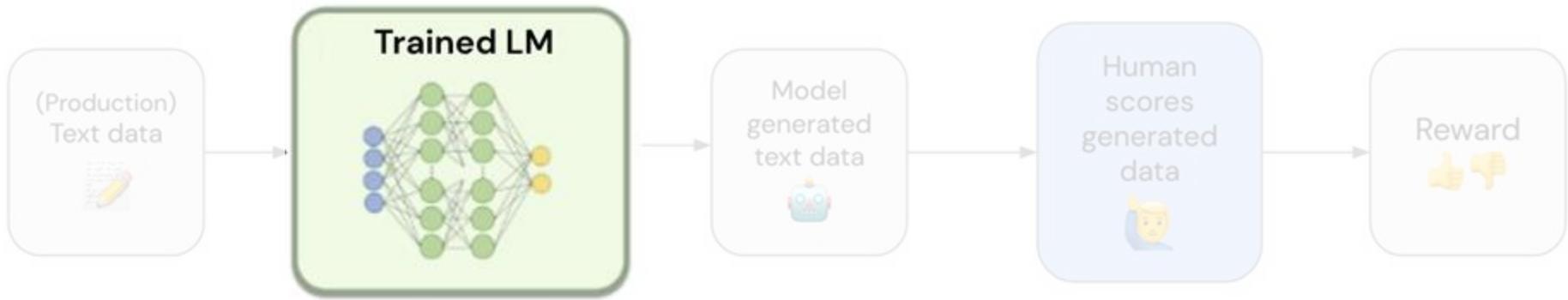
text-  
davinci-002



text-  
davinci-003

davinci-  
3.5-turbo  
(ChatGPT)

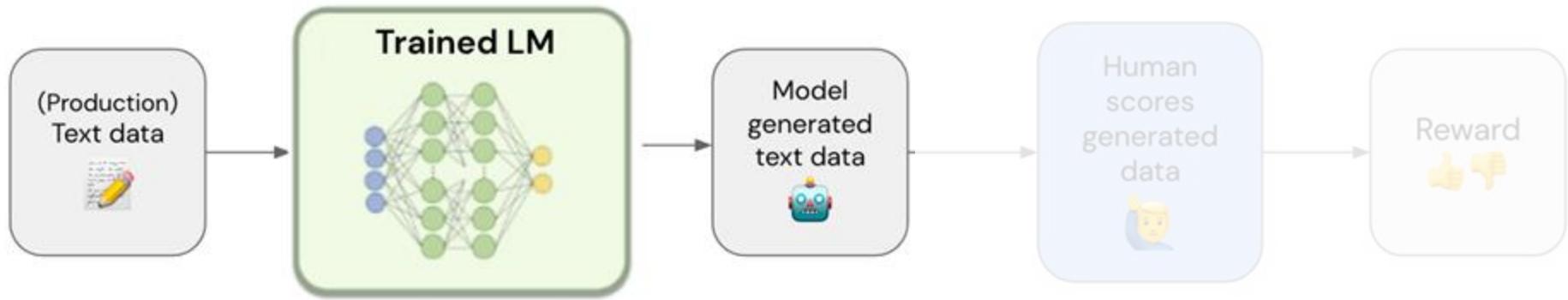
# Reinforcement Learning from Human Feedback (RLHF)



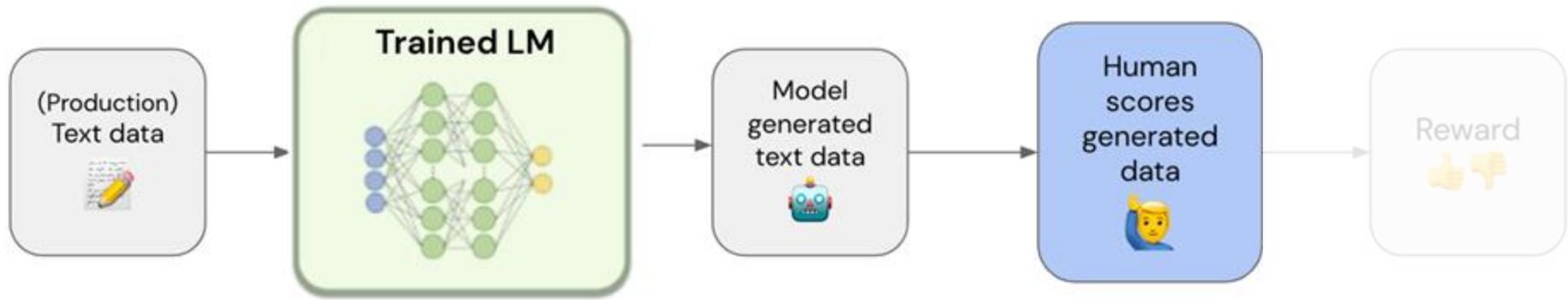
# Reinforcement Learning from Human Feedback (RLHF)



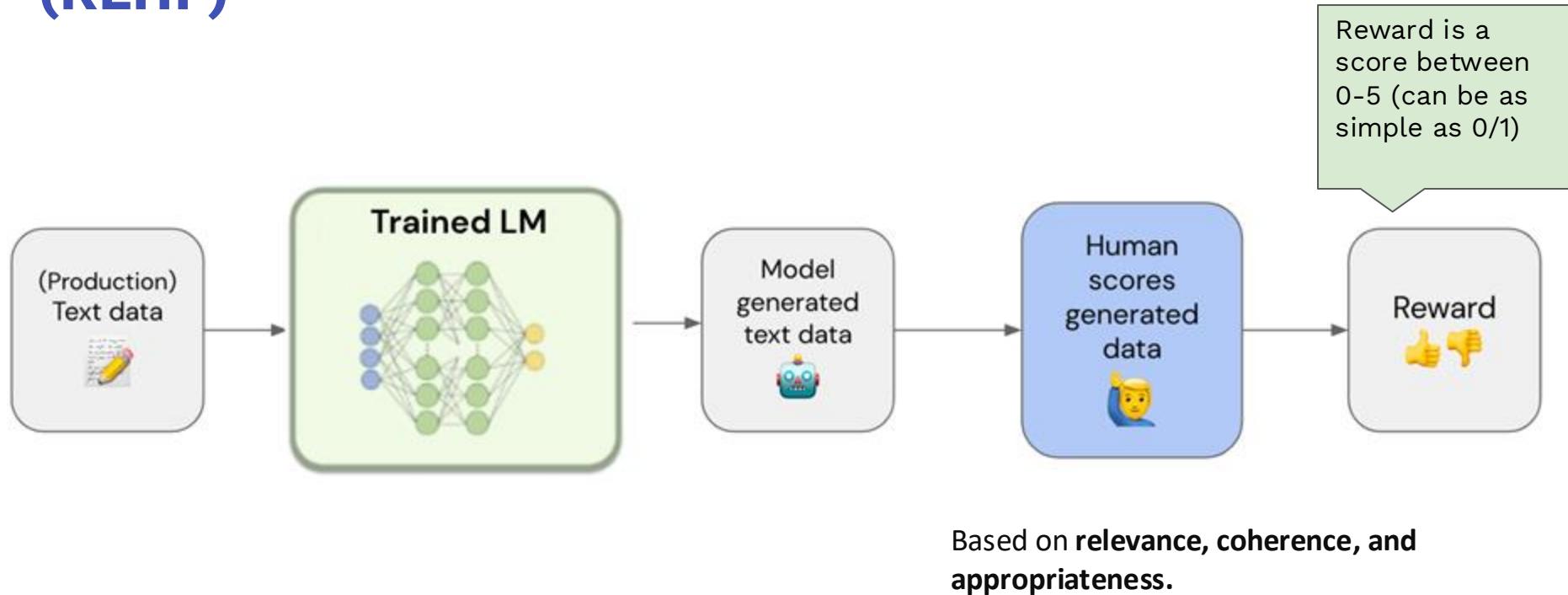
# Reinforcement Learning from Human Feedback (RLHF)



# Reinforcement Learning from Human Feedback (RLHF)



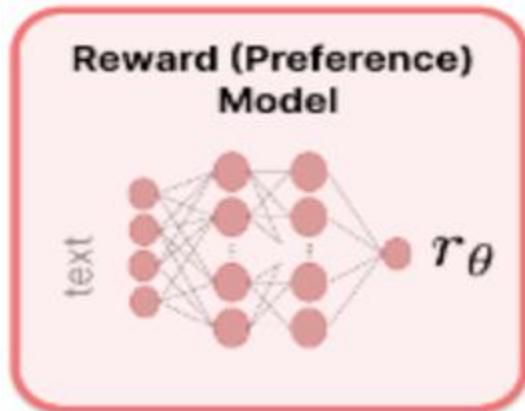
# Reinforcement Learning from Human Feedback (RLHF)



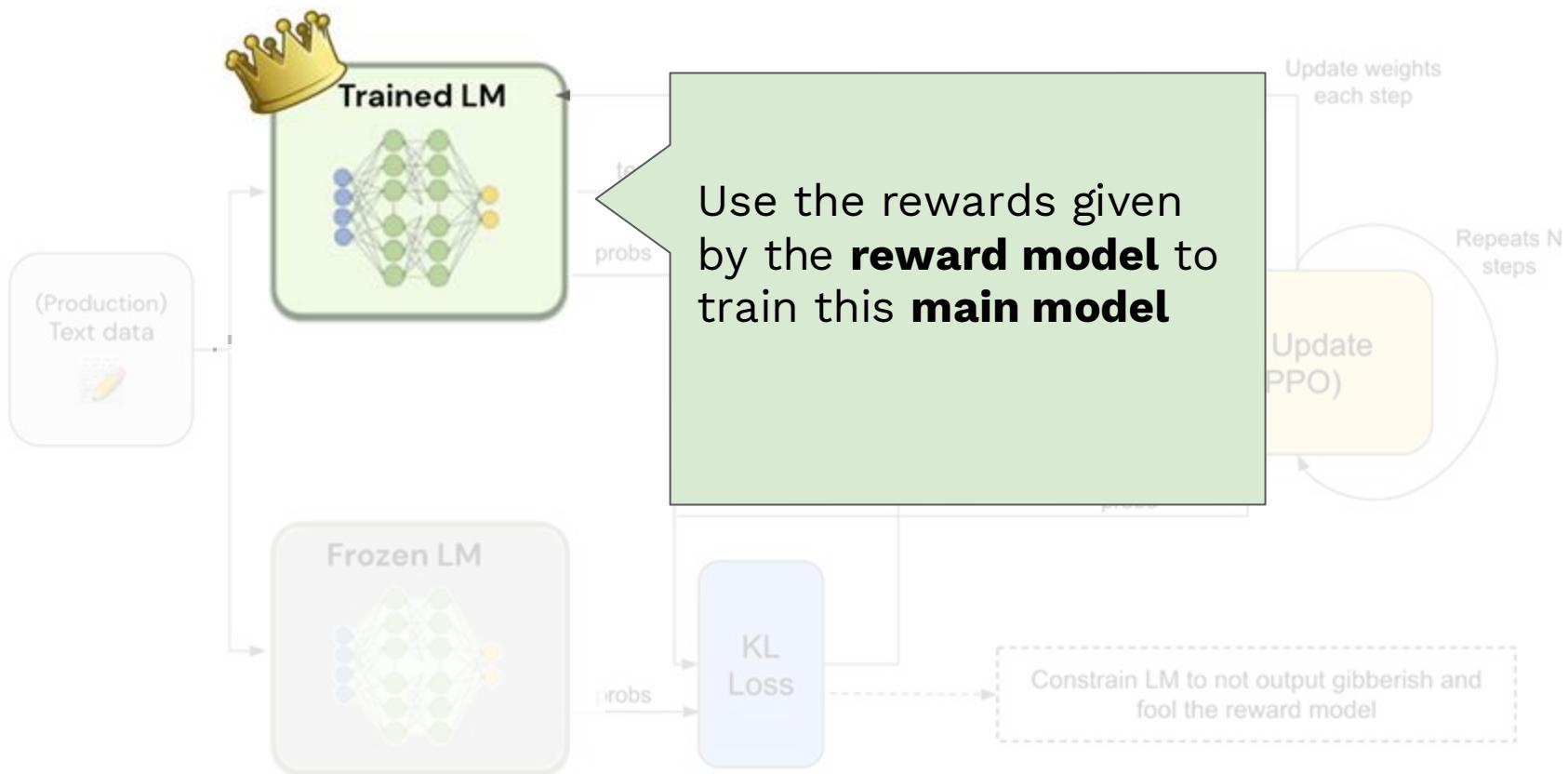
# Training a Reward Model

Using the new dataset, train another language model to receive the (input, output) text and **return a reward score**

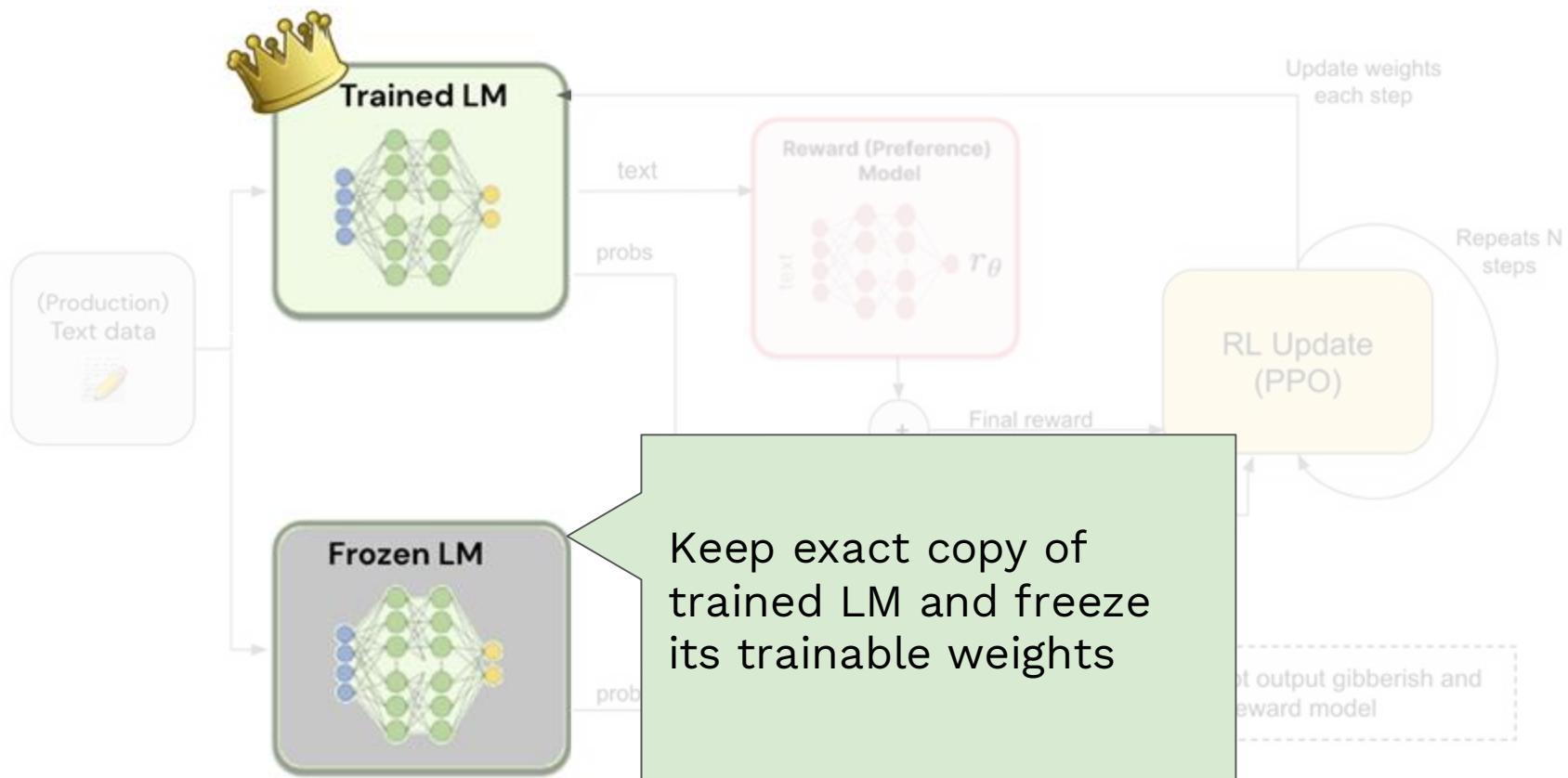
Mimic the human's reward labeling



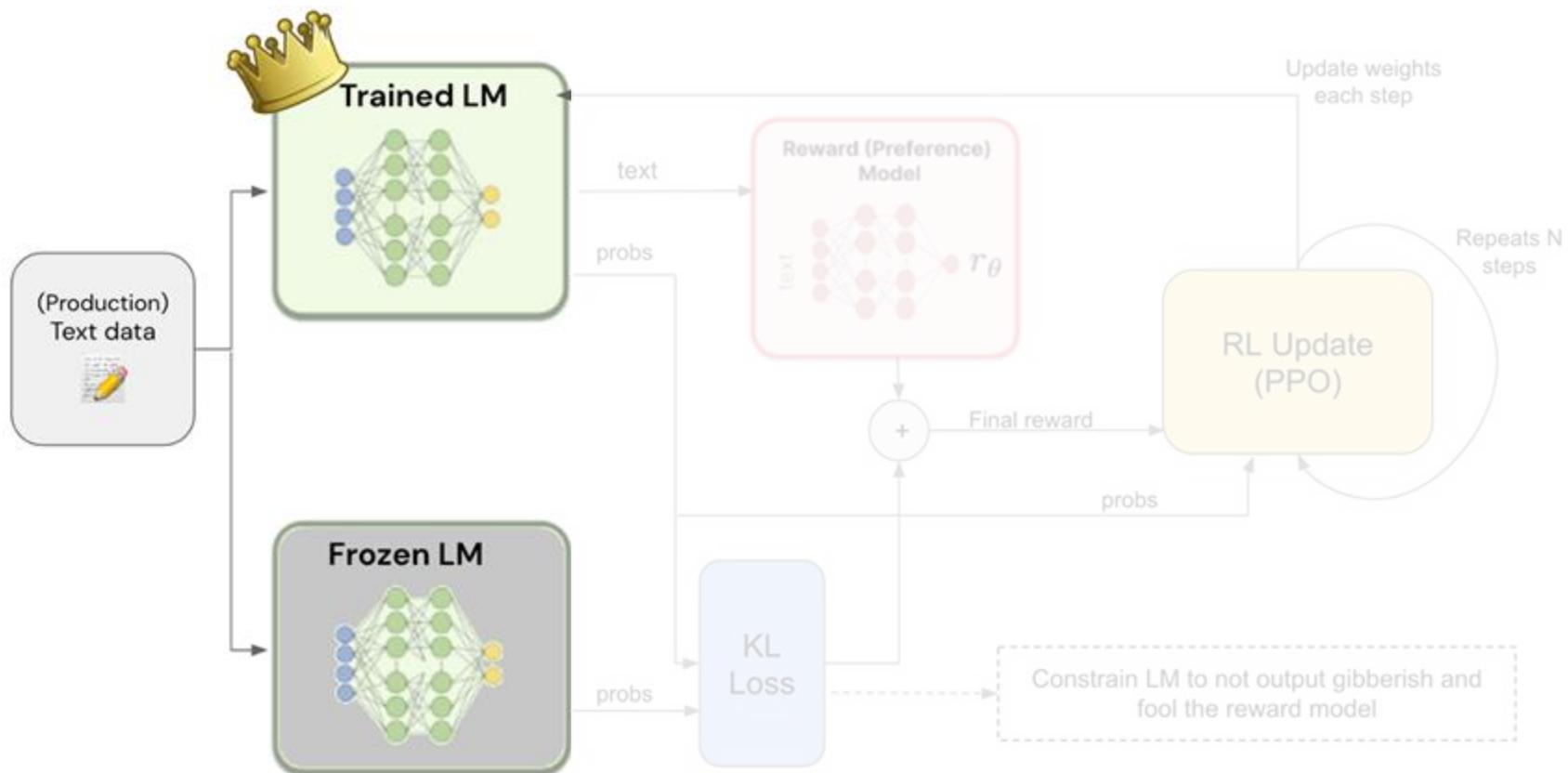
# Fine-tuning the LM with RL



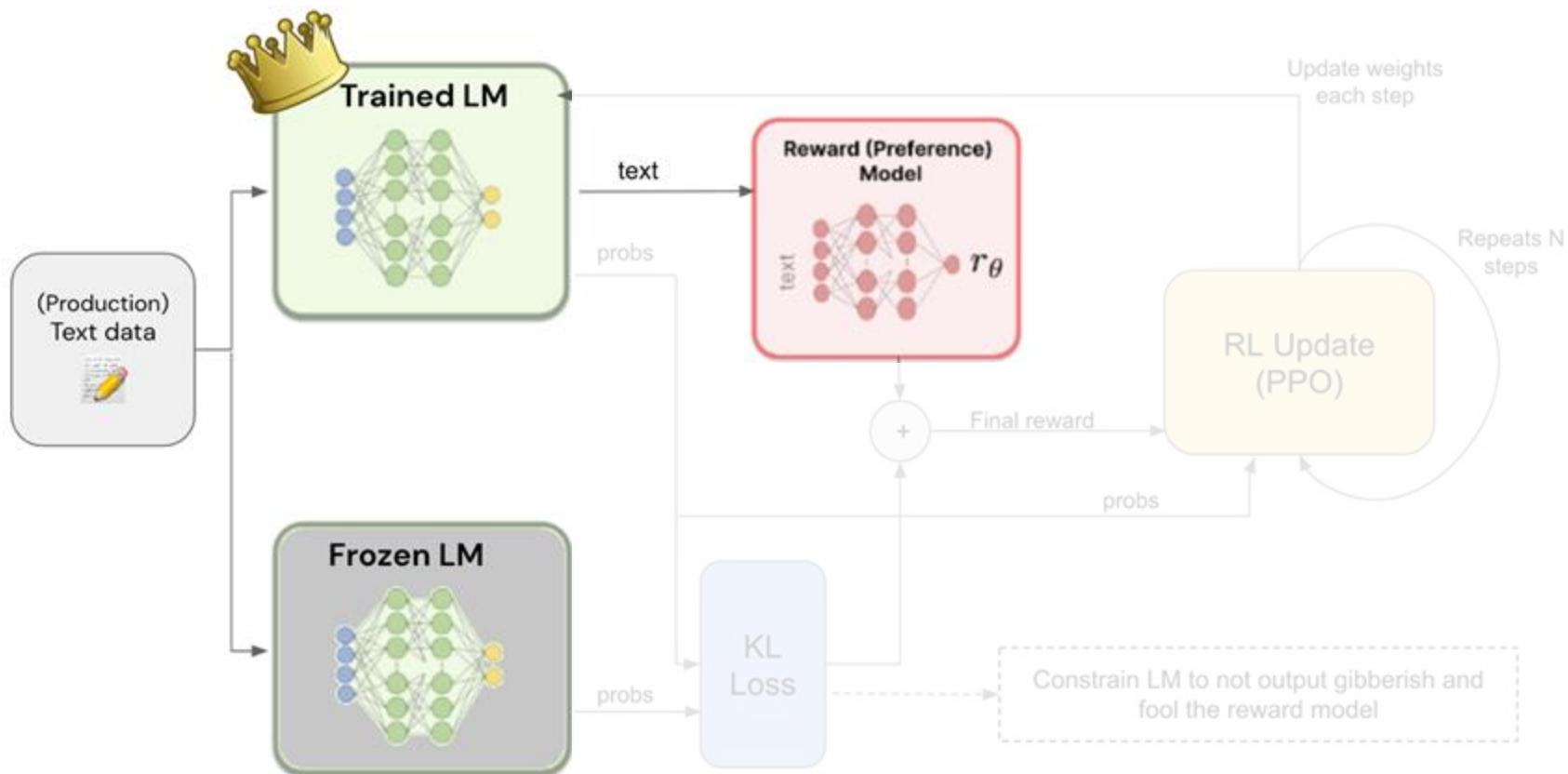
# Fine-tuning the LM with RL



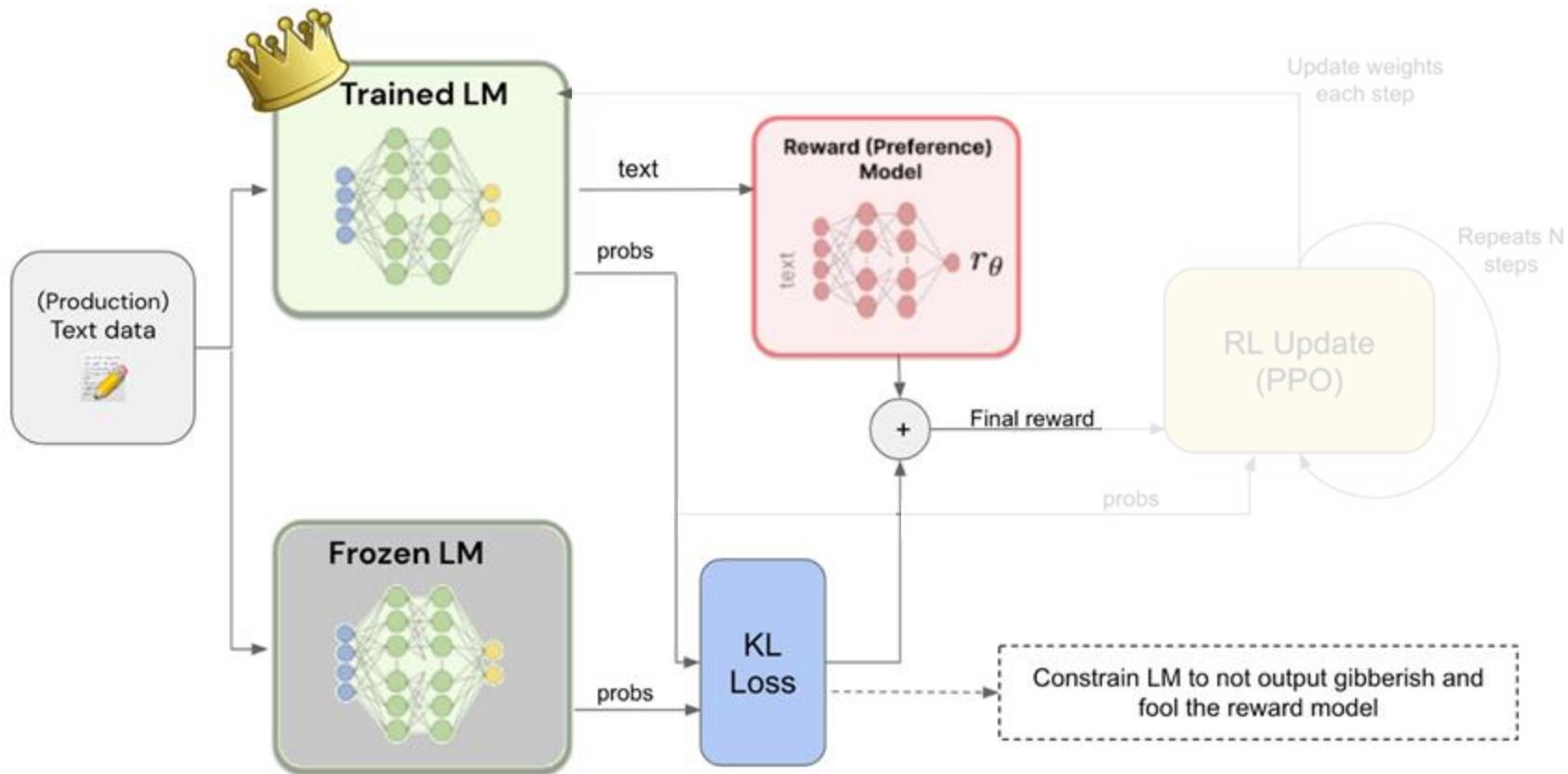
# Fine-tuning the LM with RL



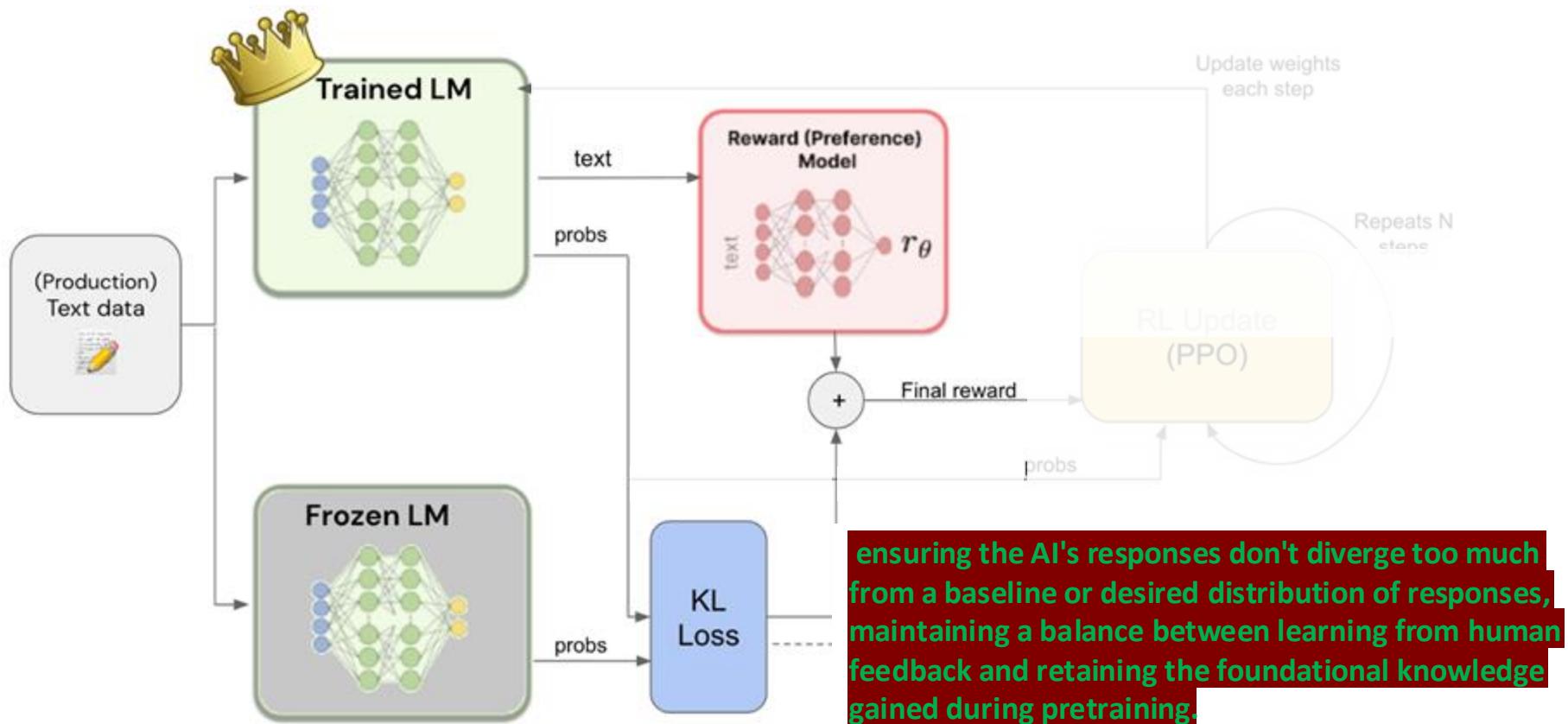
# Fine-tuning the LM with RL



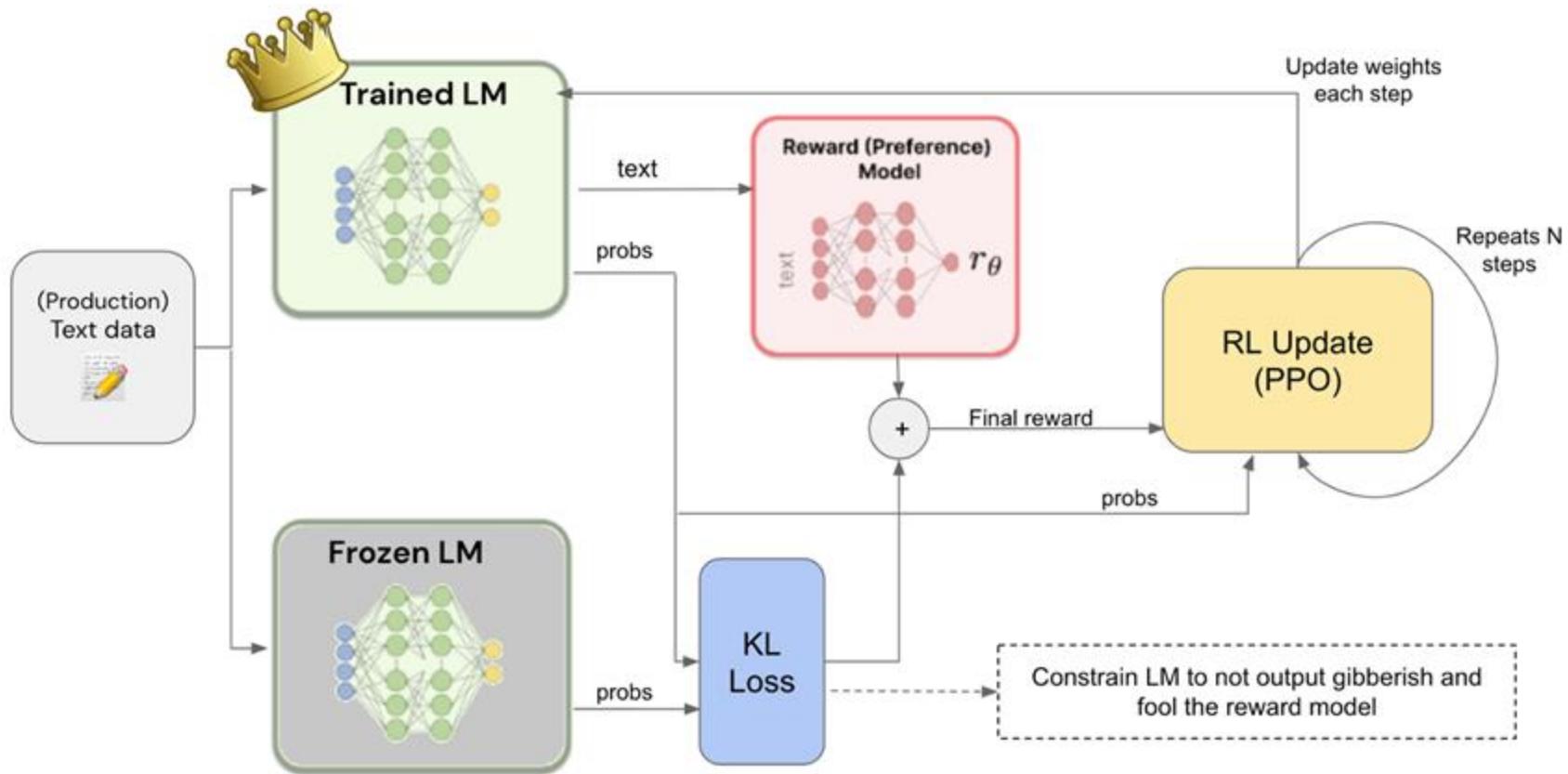
# Fine-tuning the LM with RL



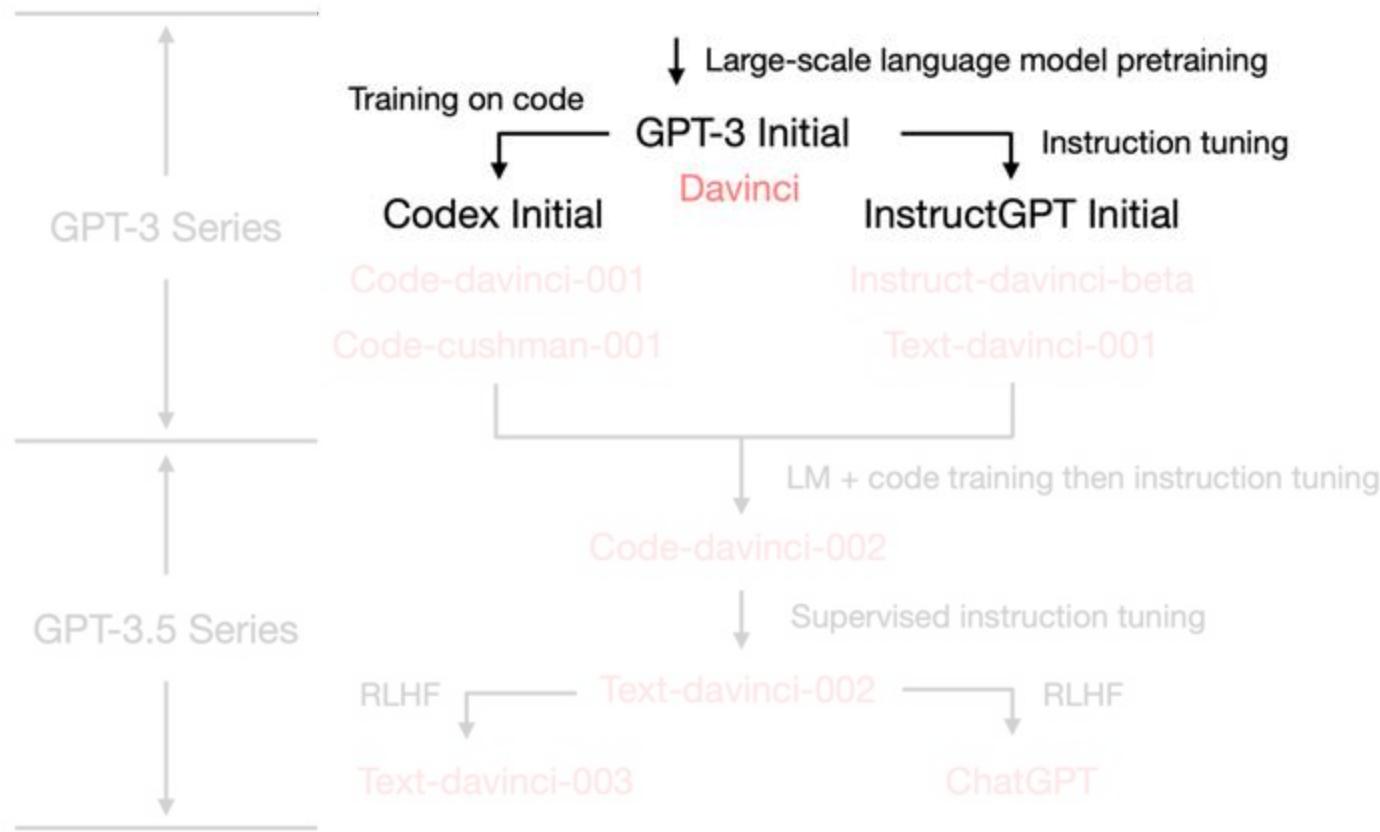
# Fine-tuning the LM with RL



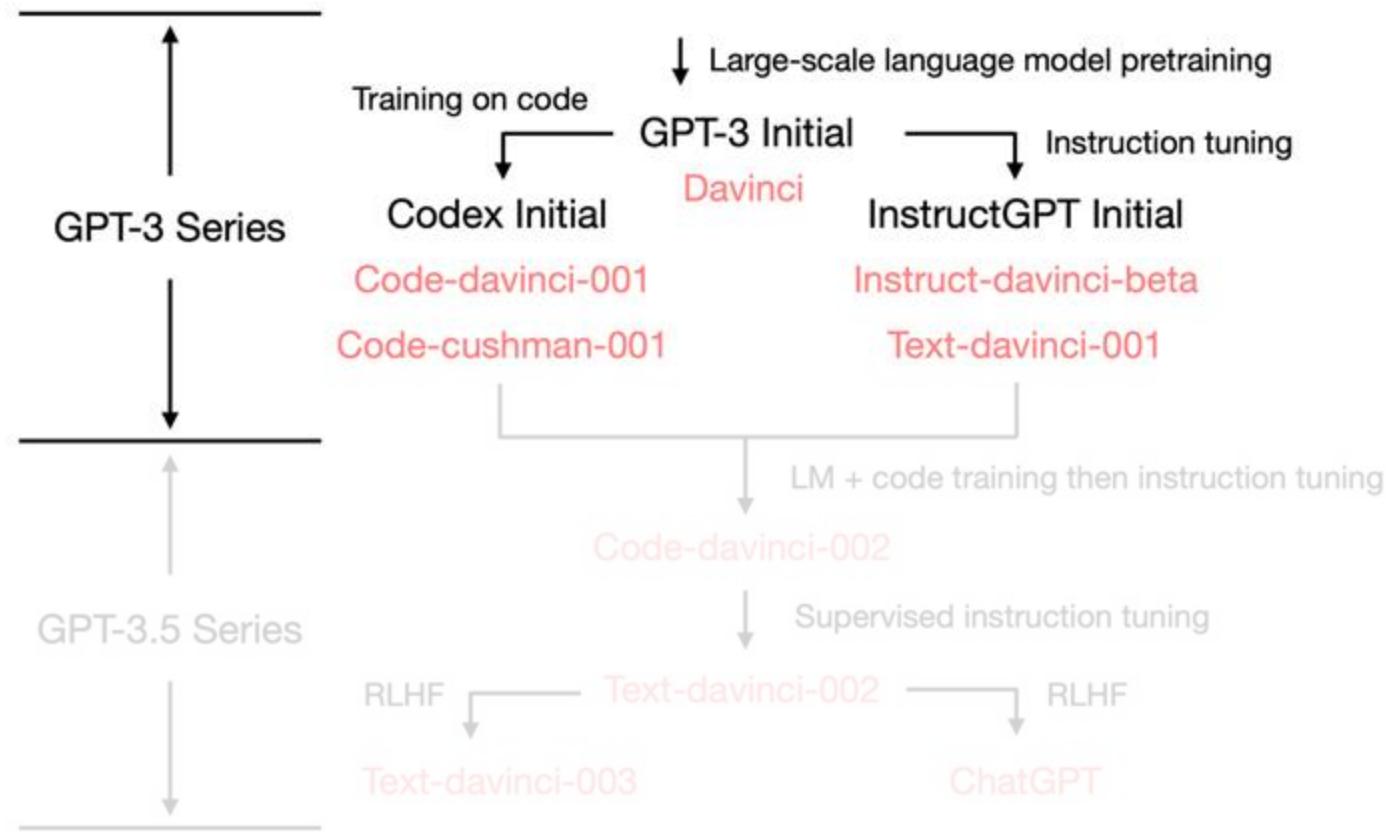
# Fine-tuning the LM with RL



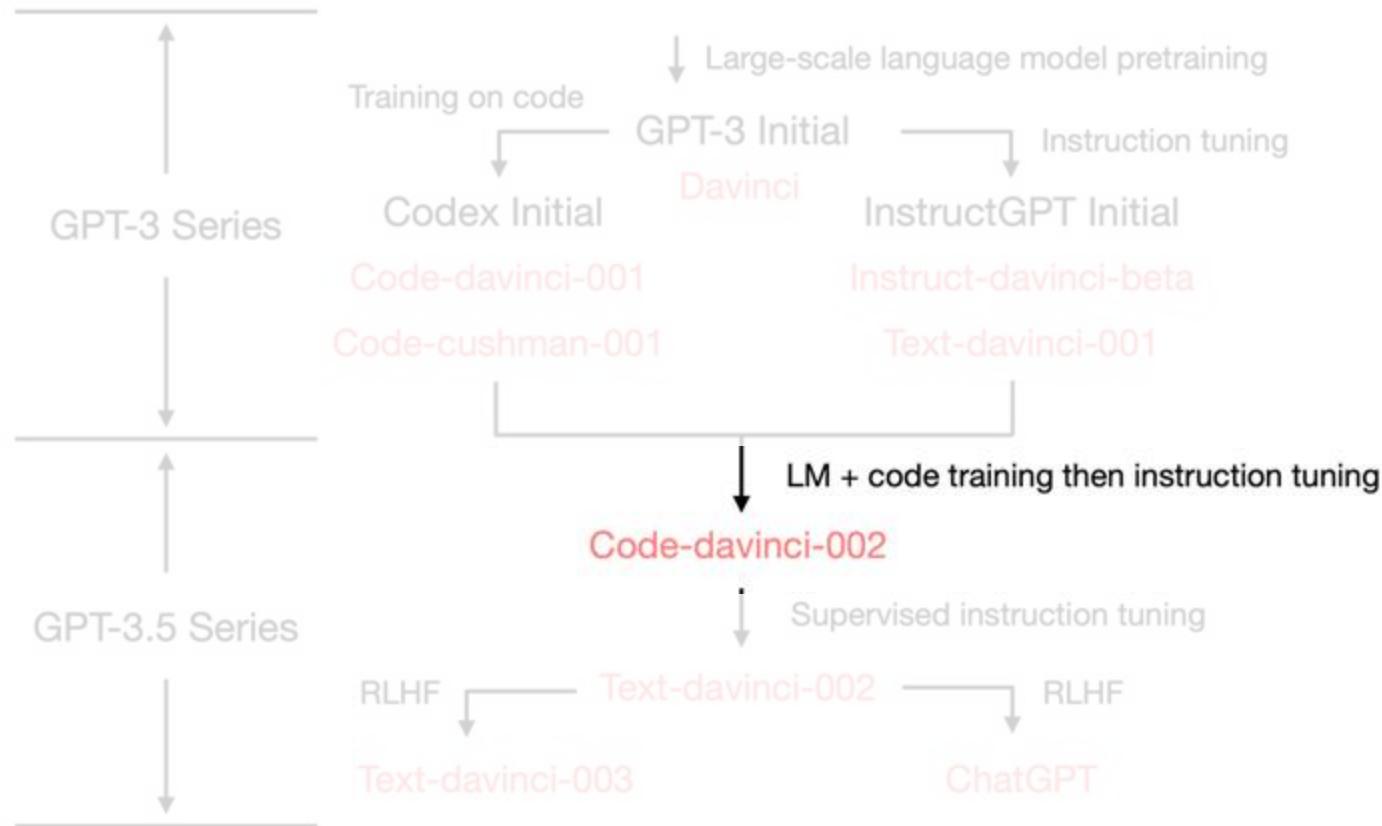
# Tracing the Origin Of Different Models



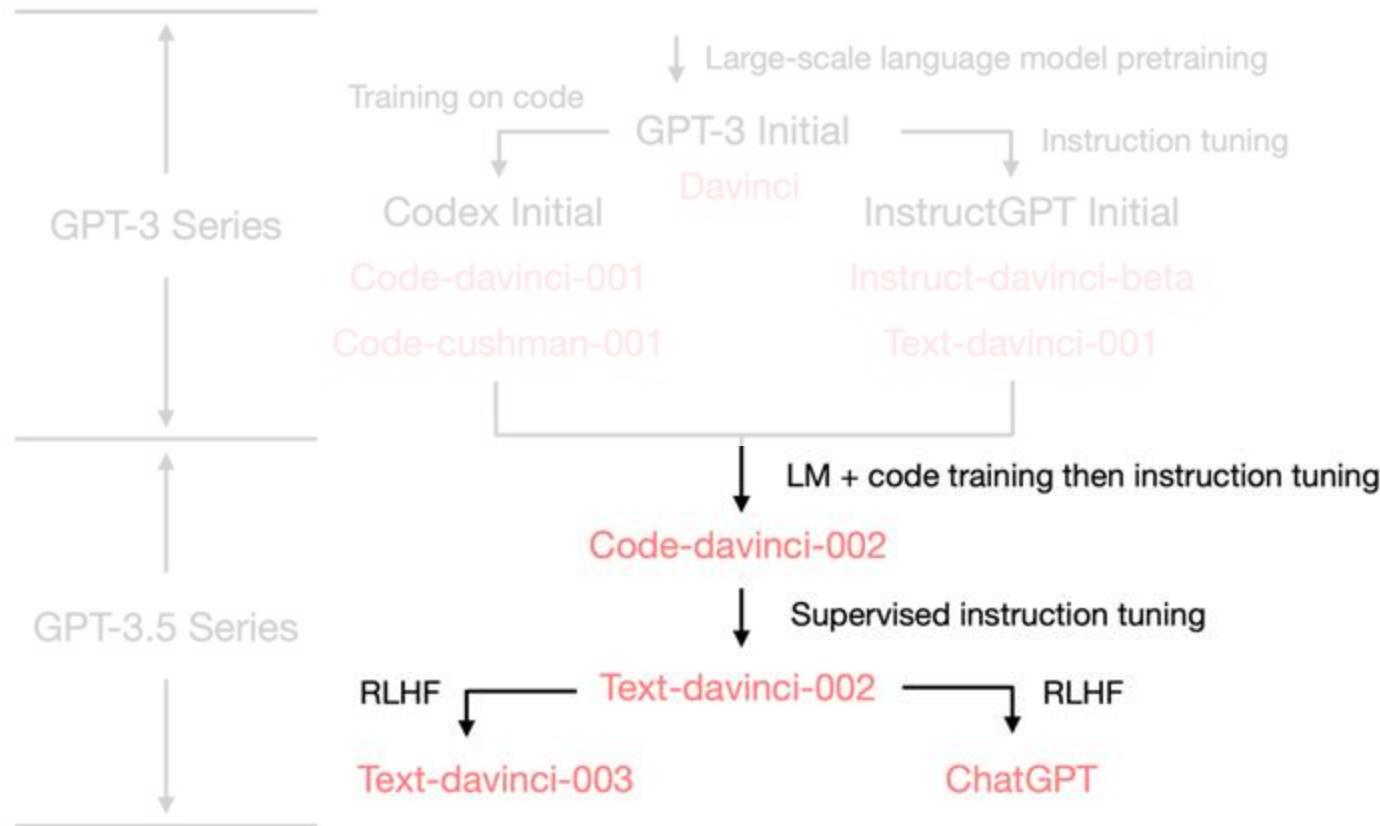
# Tracing the Origin Of Different Models



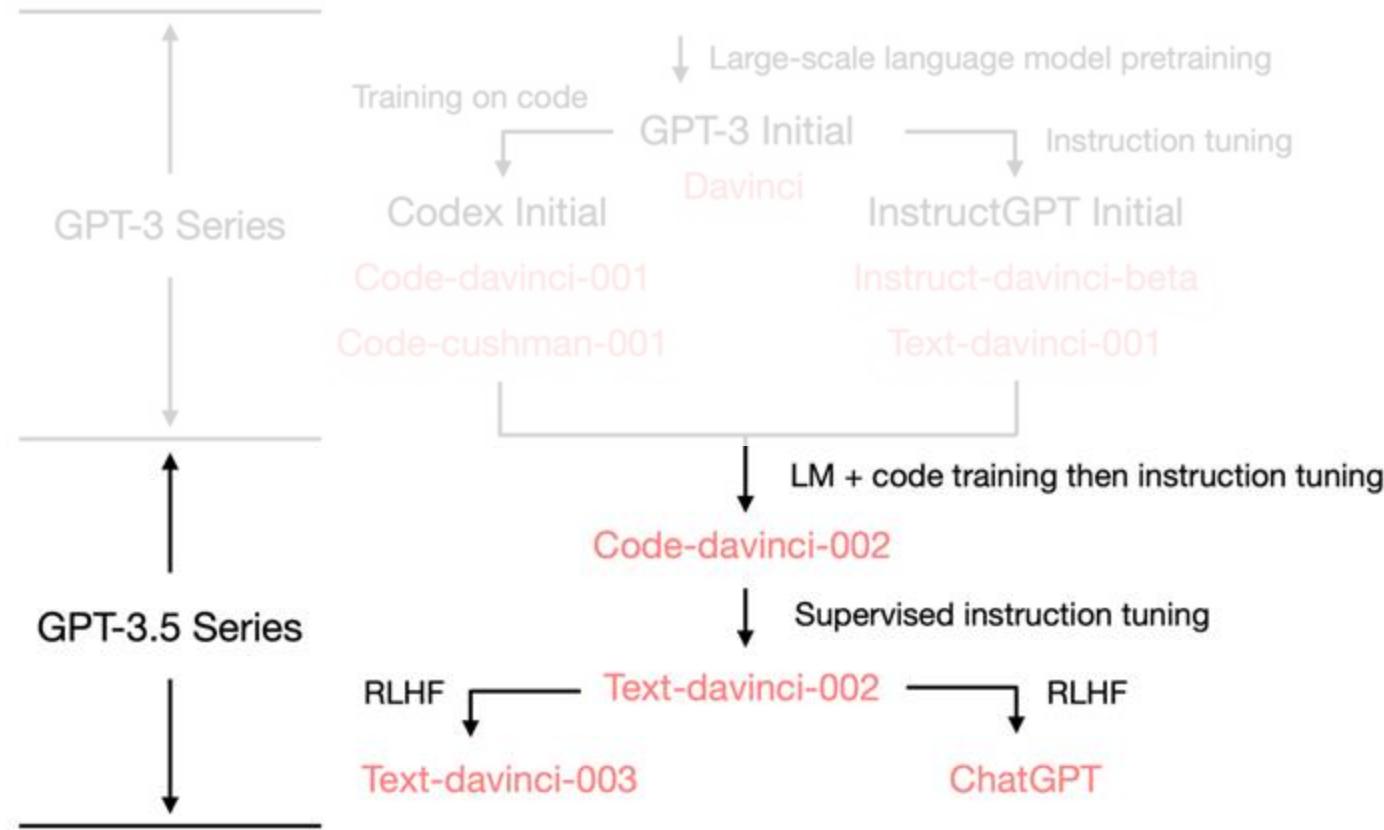
# Tracing the Origin Of Different Models



# Tracing the Origin Of Different Models



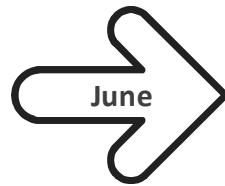
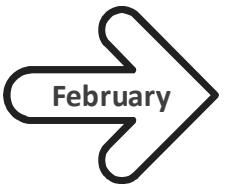
# Tracing the Origin Of Different Models



# Prominent LLMs



# LLMs - 2022



January

February

March

April

May

June

CM3  
13B  
(Meta)

GPT-NeoX  
20B  
(EleutherAI)

Chinchilla  
70B  
(DeepMind)

PaLM  
540B  
(Google)

LaMDA 2  
137B  
Chatbot  
(Google)

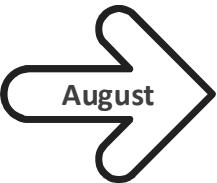
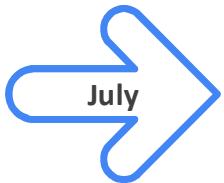
GODEL-XL  
2.7B  
Chatbot  
(Microsoft)

**BLIP**  
(Salesforce)

BLOOM  
176B  
(BigScience)

**CLIP v2**  
(OpenAI)

# LLMs - 2022



No Language  
Left Behind  
54.5B  
(Meta AI)

BlenderBot 3  
175B  
Chatbot  
(Meta AI)

Sparrow  
70B  
Chatbot  
(DeepMind)

CodeGeeX  
13B  
**Code**  
(Tsinghua)

Flan-PaLM  
540B  
(Google)

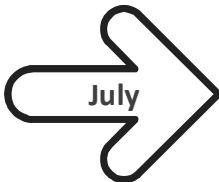
Galactica  
120B  
(Meta AI)

The Stack  
**Code dataset**  
(ServiceNow)

**ChatGPT**  
175B  
Chatbot  
(OpenAI)

ERNIE-Code  
560M  
**Code**  
(Baidu)

# LLMs - 2023



**LLaMA**

Non-commercial  
research  
(Meta AI)

**GPT-4**

Exact number  
unknown  
(OpenAI)

**PaLM 2**  
(Google)

**Llama 2**

Non-commercial  
research  
(Meta AI)

**Claude 2.1**  
(Anthropic)

**Mixtral**  
**Apache 2.0**  
(Mistral AI)

**Gemini**  
(Google)

04

## Training & Inference



# Problems with training LLMs

- Why are LLMs difficult to train?
- Write down two Challenges and potential Solutions!

# Problems with training LLMs

- **Massive amounts of data**

# Problems with training LLMs

- Massive amounts of data
- **Massive models don't fit on a single multi-GPU machine**

# Problems with training LLMs

- Massive amounts of data
- Massive models don't fit on a single multi-GPU machine
- Long training runs are painful

# Solutions?

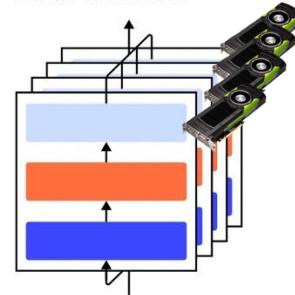
- Massive amounts of data
- Massive models don't fit on a single multi-GPU machine
- Long training runs are painful

# Parallelism

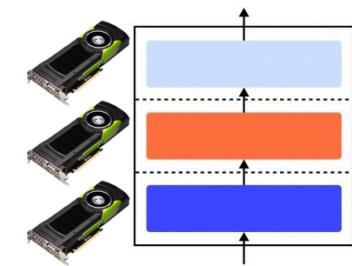
- Data parallelism: spread a single batch of data across GPUs
- Model parallelism: spread the model's layers across GPUs
- Tensor parallelism: spread a single matrix op across GPUs

(Tensor is an n-dimensional array)

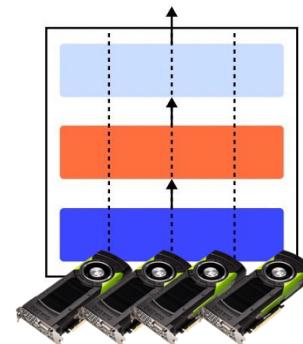
Data Parallelism



Pipeline Parallelism



Tensor Parallelism



# Problems with training LLMs

- Massive amounts of data
- Massive models don't fit on a single GPU or even a single multi-GPU machine
- **Long training runs are painful**

# A glimpse into training hell

## OPT: Open Pre-trained Transformer Language Models

Susan Zhang\*, Stephen Roller\*, Naman Goyal\*,  
Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li,  
Xi Victoria Lin, Todor Mihaylov, Myle Ott†, Sam Shleifer†, Kurt Shuster, Daniel Simig,  
Punit Singh Koura, Anjali Sridhar, Tianlu Wang, Luke Zettlemoyer

Meta AI

{susanz, roller, naman}@fb.com

- Dozens of manual restarts, **70+ automatic restarts due to HW failures**
- Manual restarting from checkpoints when **loss would diverge**
- **Switching optimizers and software versions in the middle of training**

2021-11-28 10:09am ET [Stephen]: 12.30

- 12.29 failed with the same `filename storages not found`
  - Since the exception said pg0-55, i ssh'd into it and tried manually loading its checkpoints. All 6 parts got the same storages exception!
  - I could replicate this with the storages I had manually downloaded
  - Conclusion: 33250 checkpoints are corrupt. Maybe from R12.26 and R12.25 aggressively overwriting the checkpoints.

# Some Challenges with LLMs



# What to do about the “Black Box”?

- Current LLMs are proprietary and little information is released about them, what to do?
- **Solutions?**

# What to do about the “Black Box”?

- Current LLMs are proprietary and little information is released about them, what to do?
- **Solutions?**
  - Build competitive open and well-documented models (e.g. Meta’s [OPT](#)/[LLaMa](#), Stanford’s [Alpaca](#))
  - Probe models for properties (e.g. [BIGBench](#), adversarial “[red-teaming](#)”)
  - Urge/force providers to be more transparent (e.g. [petitions](#), regulation such as the [EU AI Act](#)) <https://artificialintelligenceact.eu/>

# Big Models Require Big Pockets

Sources estimate that

1. training GPT-3 required at least **\$4,600,000**
2. running GPT-3 on a single AWS web server to cost **\$87,000** a year at minimum.

Research funding is not enough for training and running LLMs

# EU AI Act

- After three years (2021), EU approved the AI Act (March 2024).
- The law entered into force in May 2024 within EU.
- Some AI uses will get banned later
  - AI use cases that pose a **high risk** to people's **fundamental rights**, such as in **healthcare, education, and policing**.
- It will be more **obvious** when you're interacting with an AI system
  - Tech companies will be required to **label** deepfakes and AI-generated content and **notify** people when they are interacting with a chatbot or other AI system.

# EU AI Act

- Citizens can complain if they have been harmed by an AI
  - new European AI Office to coordinate compliance, implementation, and enforcement.
- AI companies will need to be more transparent
  - AI companies developing technologies in “**high risk**” sectors, such as critical infrastructure or healthcare.
  - AI companies developing large language models need to create and keep technical documentation showing how they **built** the model, how they **respect copyright law**, and publish a publicly available summary of what **training data** went into training the AI model.
  - free open-source AI models that share every detail of how the model was built, including the model’s **architecture**, **parameters**, and **weights**, are exempt from many of the obligations of the AI Act.<sup>102</sup>

# UN AI Regulation

- **On March 21, 2024, UN adopted first global AI resolution!**
- Proposed by the US and co-sponsored by China and over 120 other nations.
- Encourages countries to **safeguard** human rights, **protect** personal data, and **monitor** AI for risks.
- Nonbinding but still important.
- "The **improper** or **malicious** design, development, deployment and use of artificial intelligence systems ... pose risks that could ... undercut the protection, promotion and enjoyment of human rights and fundamental freedoms,"

<https://documents.un.org/doc/undoc/ltd/n24/065/92/pdf/n2406592.pdf?token=ZU7FDFIX3Vv8mo1Ggx&fe=true>

# AIDA Canada's AI Act

- Artificial Intelligence and Data Act (AIDA)
- Proposed in June 2022 by Government of Canada
- “ensure the development of **responsible AI** in Canada, and to prominently position Canadian firms and values in global AI development.”
- AIDA’s proposed regulations will largely apply to what it calls “**high-impact AI systems**”, essentially the same concept as the EU AI Act’s “high-risk” category.
- Not yet been passed, but coming soon...

# Reflection and Discussion

- What do you think about such AI laws and regulations?
- Pros/Cons?
- For/Against?

# Lack of Factuality, Reliability of Results

- Model outputs are fluent but sometimes they **hallucinate**: wrong, toxic, or otherwise undesirable.
- **Solutions?**

# Lack of Factuality, Reliability of Results

- Model outputs are fluent but sometimes they **hallucinate**: wrong, toxic, or otherwise undesirable.
- **Solutions?**
  - “Cite your sources” through retrieval-based models (e.g. GPT-01, [Bing search](#), [perplexity.ai](#))
  - Getting models to “know what they know” through [calibration](#)
  - [Provide better Context](#)

# Lack of Robustness

- Models are less effective when tried on new applications, domains, and languages.
- **Solutions?**
  - Methods for engineering/prompting on individual tasks
  - Training special-purpose models for domains  
(e.g. **science**: Meta [Galactica](#), **medicine**: Google [MedPaLM](#), **finance**: [BloombergGPT](#))
  - Fine-tuning on domain-specific datasets

# Introducion to LLM APIs



# Few LLM APIs

- OpenAI APIs
  - Models available: GPT-3.5 Turbo, GPT-4
  - Tasks: Completion, fine-tuning, function calling
- Anthropic
  - Models available: Claude 3
  - Tasks: Completion.
- AWS Bedrock
  - Models available: Titan, Llama, and many more

# OpenAI ChatGPT API calls

- Install OpenAI library
- Get an OpenAI API key

**API calls are as easy as invoking a REST API**

```
secret_key = 'copy and paste your key here'

import openai

# Set up your OpenAI API credentials
openai.api_key = secret_key

# output = openai.Model.list()

output = openai.Completion.create(
    model="text-curie-001",
    prompt="tell me a joke",
)

print(output)
```

# Introduction to Running LLMs Locally



# Running LLMs Locally

- There are open-source ecosystems offering collections of chatbot LLMs similar to ChatGPT, which can be run locally on your computer.
- Why?
  - Offline Mode
  - Privacy and Security
  - Cost

# GPT4ALL

- [GPT4All](#) is an LLM that you can install on your computer
- Try different models locally with UI setting

What models are supported by the GPT4All ecosystem?

Currently, there are six different model architectures that are supported:

1. GPT-J - Based off of the GPT-J architecture with examples found [here](#)
2. LLaMA - Based off of the LLaMA architecture with examples found [here](#)
3. MPT - Based off of Mosaic ML's MPT architecture with examples found [here](#)
4. Replit - Based off of Replit Inc.'s Replit architecture with examples found [here](#)
5. Falcon - Based off of TII's Falcon architecture with examples found [here](#)
6. StarCoder - Based off of BigCode's StarCoder architecture with examples found [here](#)

**Start With  
GPT-J**

# Libraries to Develop LLM Applications



# Popular libraries for LLM Ecosystem

- LangChain
  - Popular Python library for chaining multiple generative models together
- Streamlit
  - For building ChatGPT like web interface
- Hugging Face
  - Python tools for pre-processing, training, fine-tuning, and deployment
- Pinecone, Chroma DB, Milvus
  - Vector databases

# AI Feature A6

You will need to integrate an AI-based feature into your web application.



# Questions?

