

Large Language Models (LLMs)

Michael T. Goodrich
University of California, Irvine

Slides from Sachin Kumar, Roshan Sharma, and Dan Jurafsky

The language modeling problem

Rank these sentences in the order of plausibility?

1. Jane went to the store.
2. store to Jane went the.
3. Jane went store.
4. Jane goed to the store.
5. The store went to Jane.
6. The food truck went to Jane.

How probable is a piece of text? Or what is $p(\text{text})$

$$p(\text{how are you this evening ? has your house ever been burgled ?}) = 10^{-15}$$

$$p(\text{how are you this evening ? fine , thanks , how about you ?}) = 10^{-9}$$

The language modeling problem

A **language model** answers the question: **What is $p(\text{text})$?**

Text is a sequence of symbols: $(x^{(1)}, x^{(2)}, \dots, x^{(N)})$

$$p(x^{(1)}, x^{(2)}, \dots, x^{(N)})$$

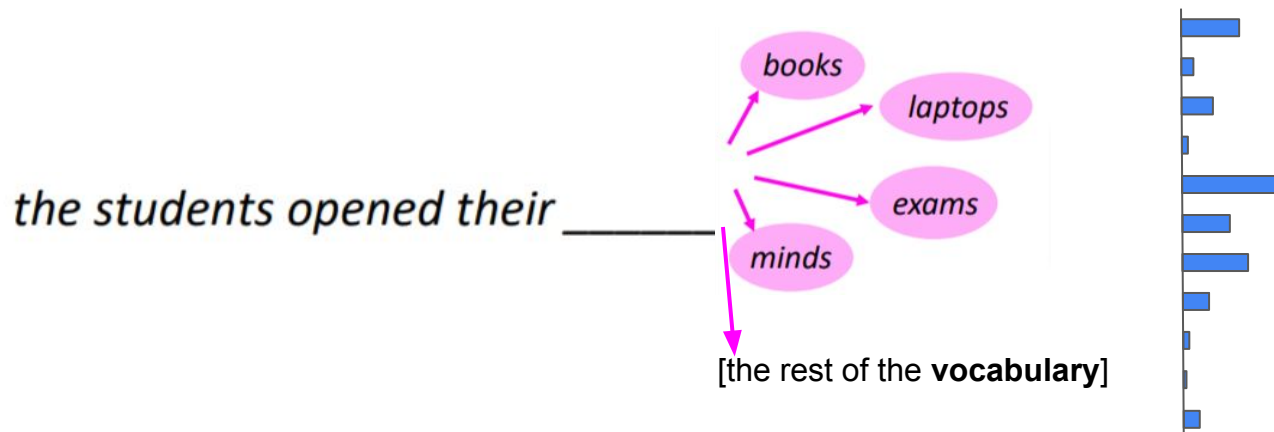
$$p(x^{(1)})p(x^{(2)}|x^{(1)})p(x^{(3)}|x^{(1)}, x^{(2)}) \dots$$

$$\prod_{i=1}^N p(x^{(i)} | \underbrace{x^{(1)}, \dots, x^{(i-1)}}_{\text{context}})$$

Just the chain rule of probability— no simplifying assumptions!

The language modeling problem

$$\prod_{i=1}^N p(x^{(i)} | \underbrace{x^{(1)}, \dots, x^{(i-1)}}_{\text{context}})$$



Language models of this form can generate text

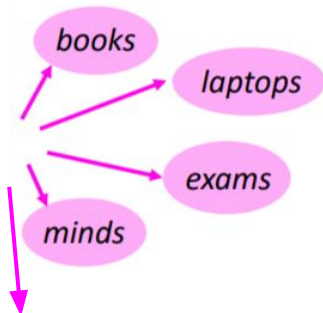
At each timestep, sample a token from the language model's new probability distribution over next tokens.

The _____

The students _____

The students opened _____

The students opened their _____



Language models play the role of ...


- a judge of **grammaticality**
 - e.g., should prefer “The boy runs.” to “The boy run.”
- a judge of **semantic plausibility**
 - e.g., should prefer “The woman spoke.” to “The sandwich spoke.”
- an enforcer of **stylistic consistency**
 - e.g., should prefer “Hello, how are you this evening? Fine, thanks, how are you?” to “Hello, how are you this evening? Has your house ever been burgled?”
- a repository of **knowledge (?)**
 - e.g., “Barack Obama was the 44th President of the United States”



Note that this is very difficult to guarantee!

We use language models every day



what is the | 

what is the **weather**
what is the **meaning of life**
what is the **dark web**
what is the **xfl**
what is the **doomsday clock**
what is the **weather today**
what is the **keto diet**
what is the **american dream**
what is the **speed of light**
what is the **bill of rights**

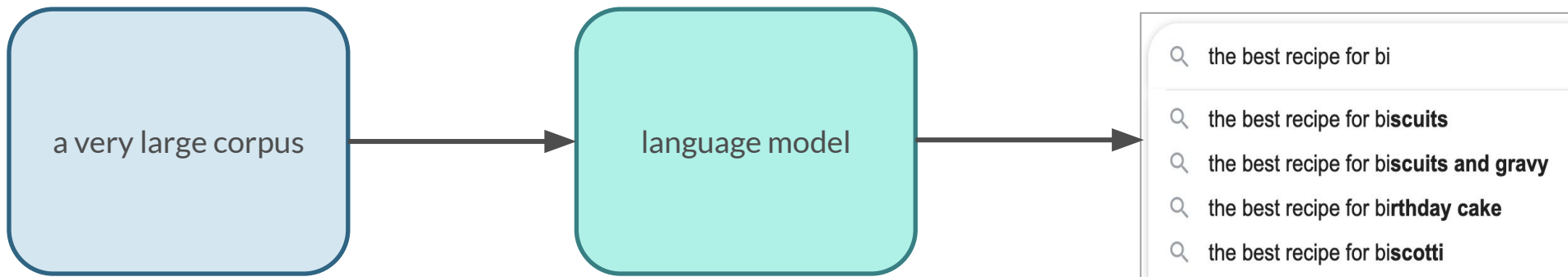
[Google Search](#) [I'm Feeling Lucky](#)

Why language modeling?

- Machine translation
 - $p(\text{strong winds}) > p(\text{large winds})$
- Spelling correction
 - The office is about fifteen minuets from my house
 - $p(\text{about fifteen minutes from}) > p(\text{about fifteen minuets from})$
- Speech recognition
 - $p(\text{I saw a van}) \gg p(\text{eyes awe of an})$
- Summarization, question-answering, handwriting recognition, OCR, etc.

How we learn a language model

Language modeling



How do we learn a language model?

Estimate probabilities using text data

- Collect a textual corpus
- Find a distribution that maximizes the probability of the corpus – maximum likelihood estimation

A naive solution: count and divide

- Assume we have N training sentences
- Let x_1, x_2, \dots, x_n be a sentence, and $c(x_1, x_2, \dots, x_n)$ be the number of times it appeared in the training data.
- Define a language model:

$$p(x_1, \dots, x_n) = \frac{c(x_1, \dots, x_n)}{N}$$

No generalization!

Markov assumption

- We make the **Markov assumption**: $\mathbf{x}^{(t+1)}$ depends only on the preceding **n-1** words

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(t+1)} | \underbrace{\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}}_{\text{n-1 words}})$$

assumption

Markov assumption

$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$

or maybe even

$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$

n-gram Language Models

$$\prod_{i=1}^N p(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

- **Question:** How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn an *n-gram* Language Model!
- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word

unigram probability

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

- corpus size $m = 17$
- $P(\text{Lucy}) = 2/17$; $P(\text{cats}) = 1/17$

- Unigram probability:
$$P(w) = \frac{\text{count}(w)}{m} = \frac{C(w)}{m}$$

bigram probability

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

$$P(A | B) = \frac{P(A,B)}{P(B)}$$

$$P(\text{have} | I) = \frac{P(I \text{ have})}{P(I)} = \frac{2}{2} = 1$$

$$P(\text{two} | \text{have}) = \frac{P(\text{have two})}{P(\text{have})} = \frac{1}{2} = 0.5$$

$$P(\text{eating} | \text{have}) = \frac{P(\text{have eating})}{P(\text{have})} = \frac{0}{2} = 0$$

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{\sum_w C(w_1, w)} = \frac{C(w_1, w_2)}{C(w_1)}$$

trigram probability

“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

$$P(A | B) = \frac{P(A,B)}{P(B)}$$

$$P(a | \text{I have}) = \frac{C(\text{I have a})}{C(\text{I have})} = \frac{1}{2} = 0.5$$

$$P(w_3 | w_1 w_2) = \frac{C(w_1, w_2, w_3)}{\sum_w C(w_1, w_2, w)} = \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)}$$

$$P(\text{several} | \text{I have}) = \frac{C(\text{I have several})}{C(\text{I have})} = \frac{0}{2} = 0$$

n-gram probability

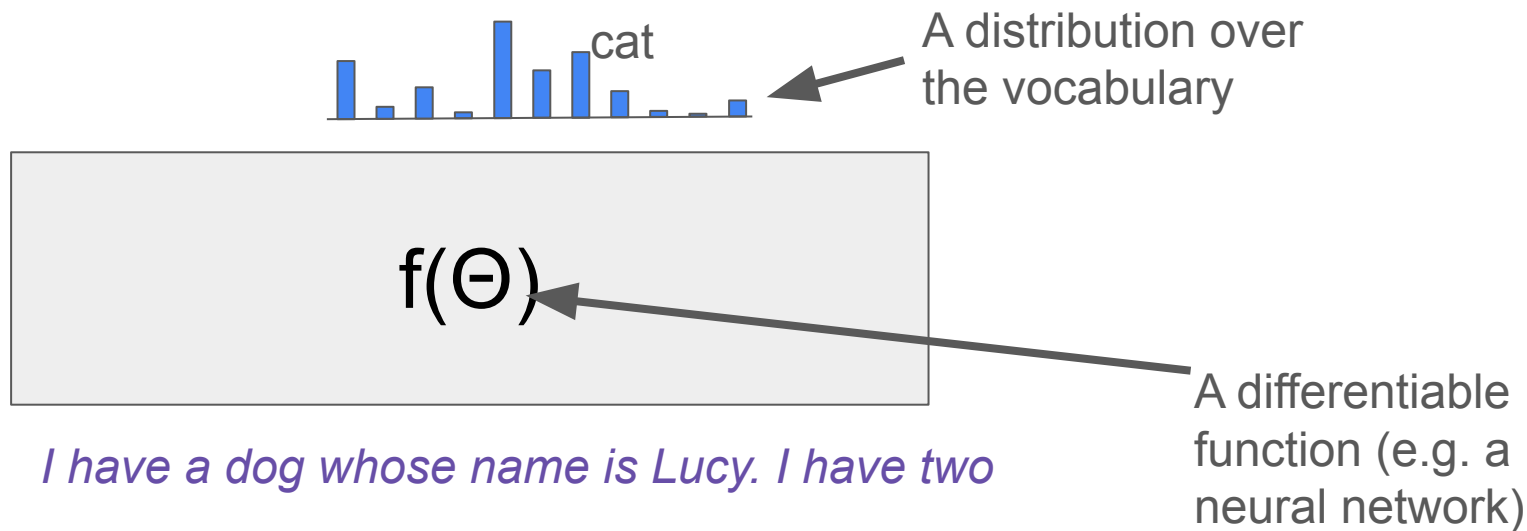
“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

$$P(w_i \mid w_1, w_2, \dots, w_{i-1}) = \frac{C(w_1, w_2, \dots, w_{i-1}, w_i)}{C(w_1, w_2, \dots, w_{i-1})}$$

Neural language models

$$\prod_{i=1}^N p(x^{(i)} | x^{(1)}, \dots, x^{(i-1)})$$



How do we maximize the likelihood?

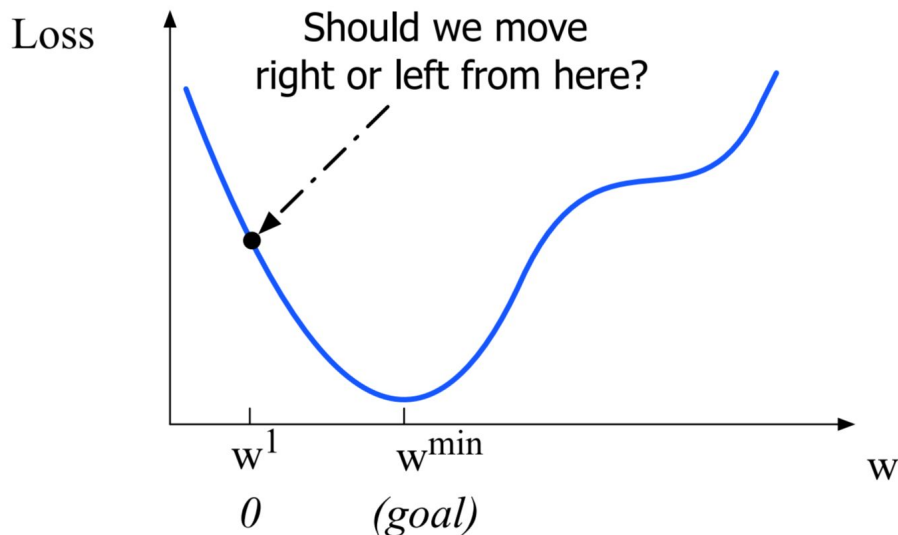
The dominant strategy from the past decade:

1. The randomly initialized differentiable function (neural network) takes the context as **input**
2. Have that function output a probability distribution over the vocabulary
3. Treat the **probability** of the correct token as your objective to maximize.
4. Or **negative log (probability)** as your objective to **minimize**
5. Differentiate with respect to the parameters, and perform **gradient descent, or**

Gradient descent: a throwback to calculus

Q: Given current parameter w , should we make w bigger or smaller to minimize our loss?

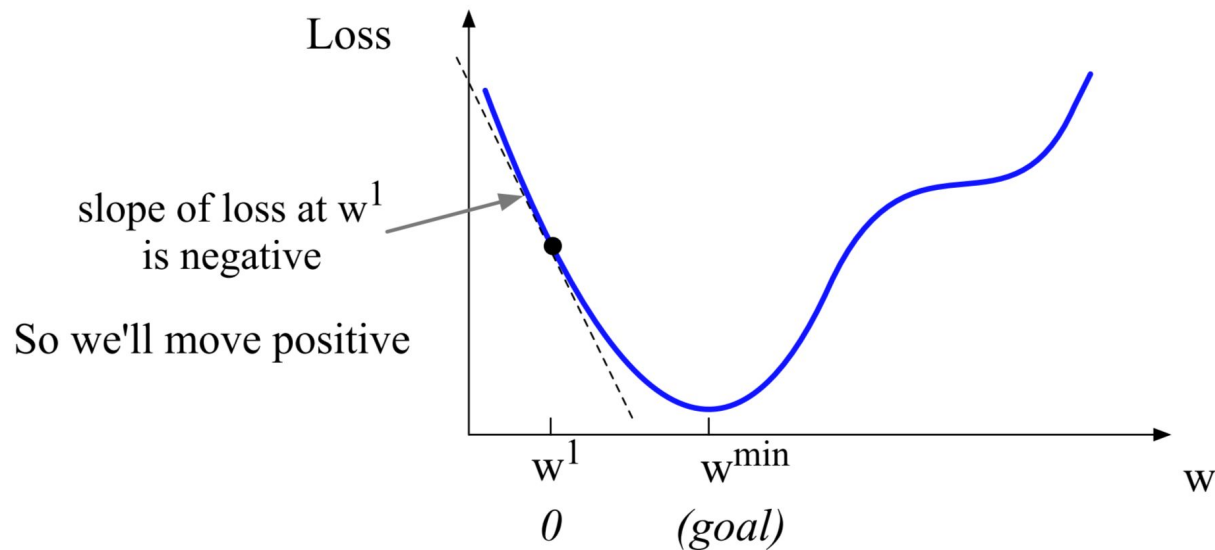
A: Move w in the reverse direction from the slope of the function



Let's first visualize for a single scalar w

Q: Given current w , should we make it bigger or smaller?

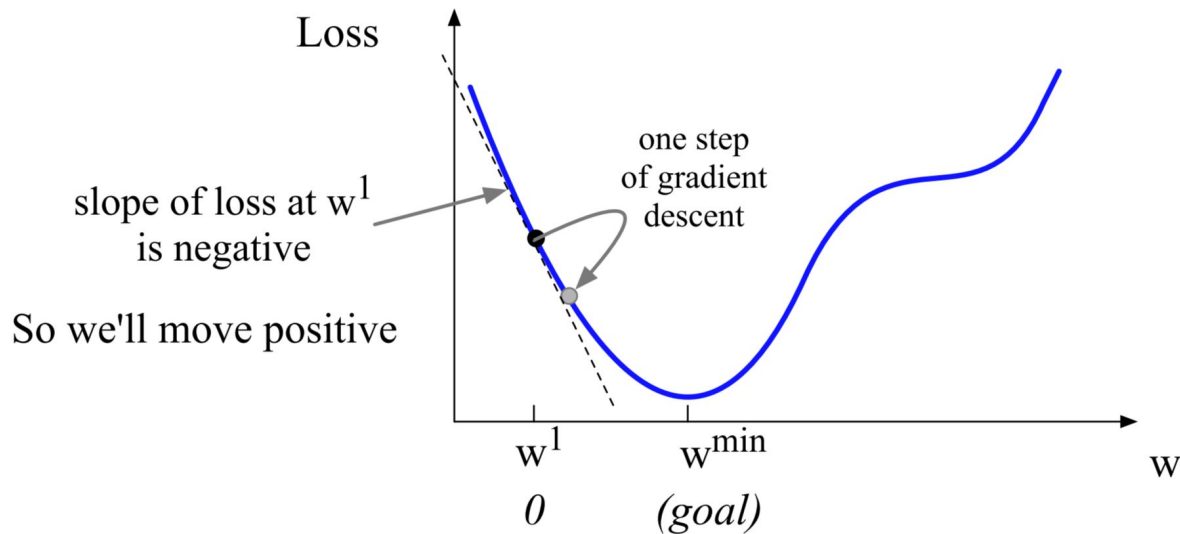
A: Move w in the reverse direction from the slope of the function



Let's first visualize for a single scalar w

Q: Given current w , should we make it bigger or smaller?

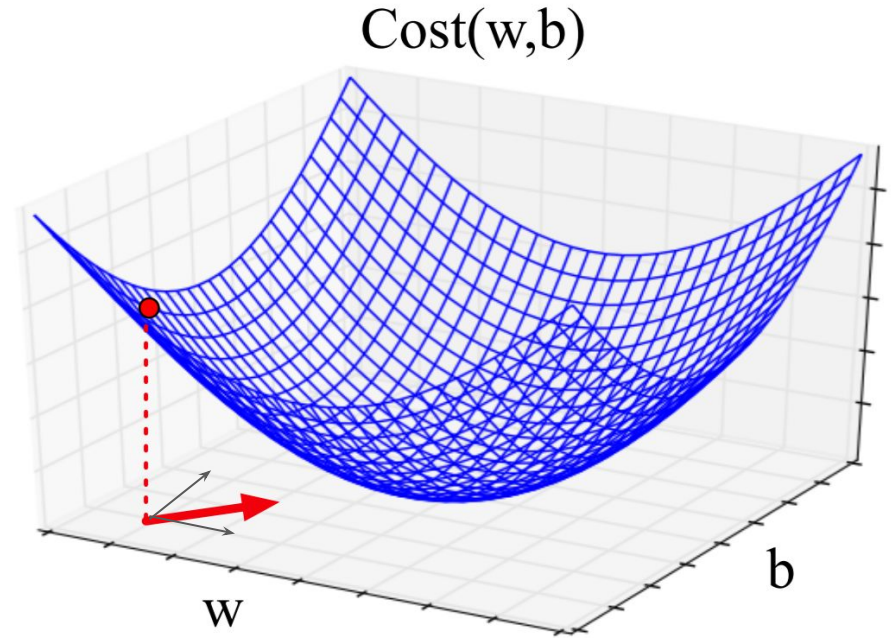
A: Move w in the reverse direction from the slope of the function



Now let's imagine 2 dimensions, w and b

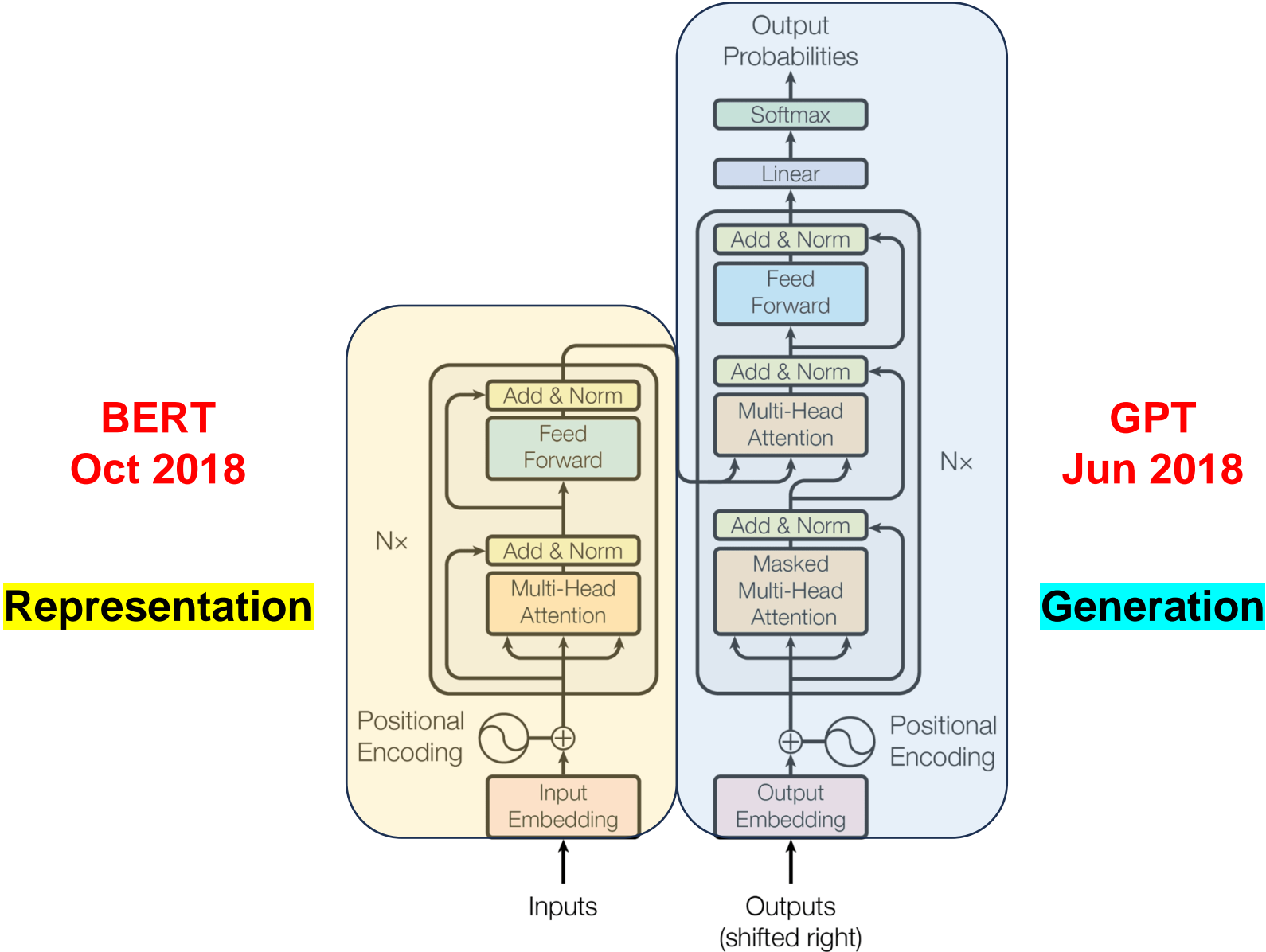
Visualizing the (negative) gradient vector
at the red point

It has two dimensions shown
in the x - y plane



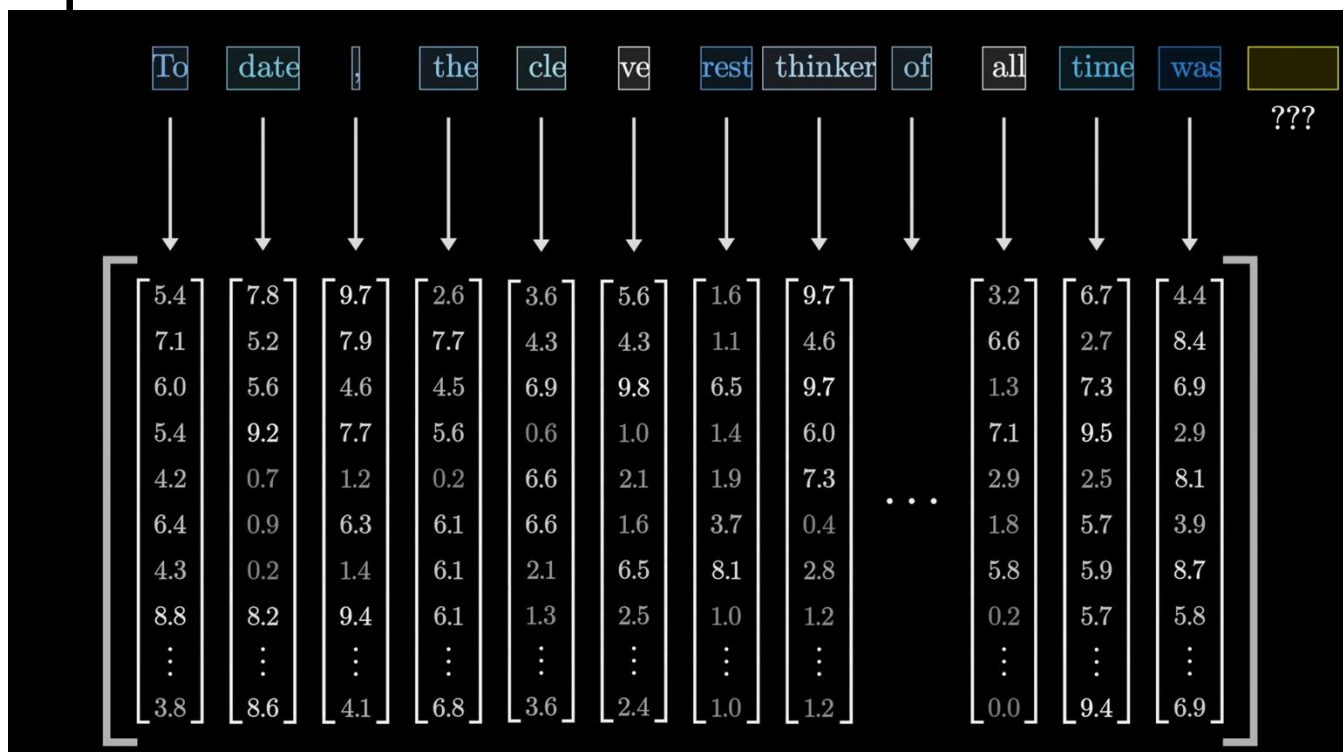
✨✨ The Transformer ✨✨

The LLM Era – Paradigm Shift in Machine Learning

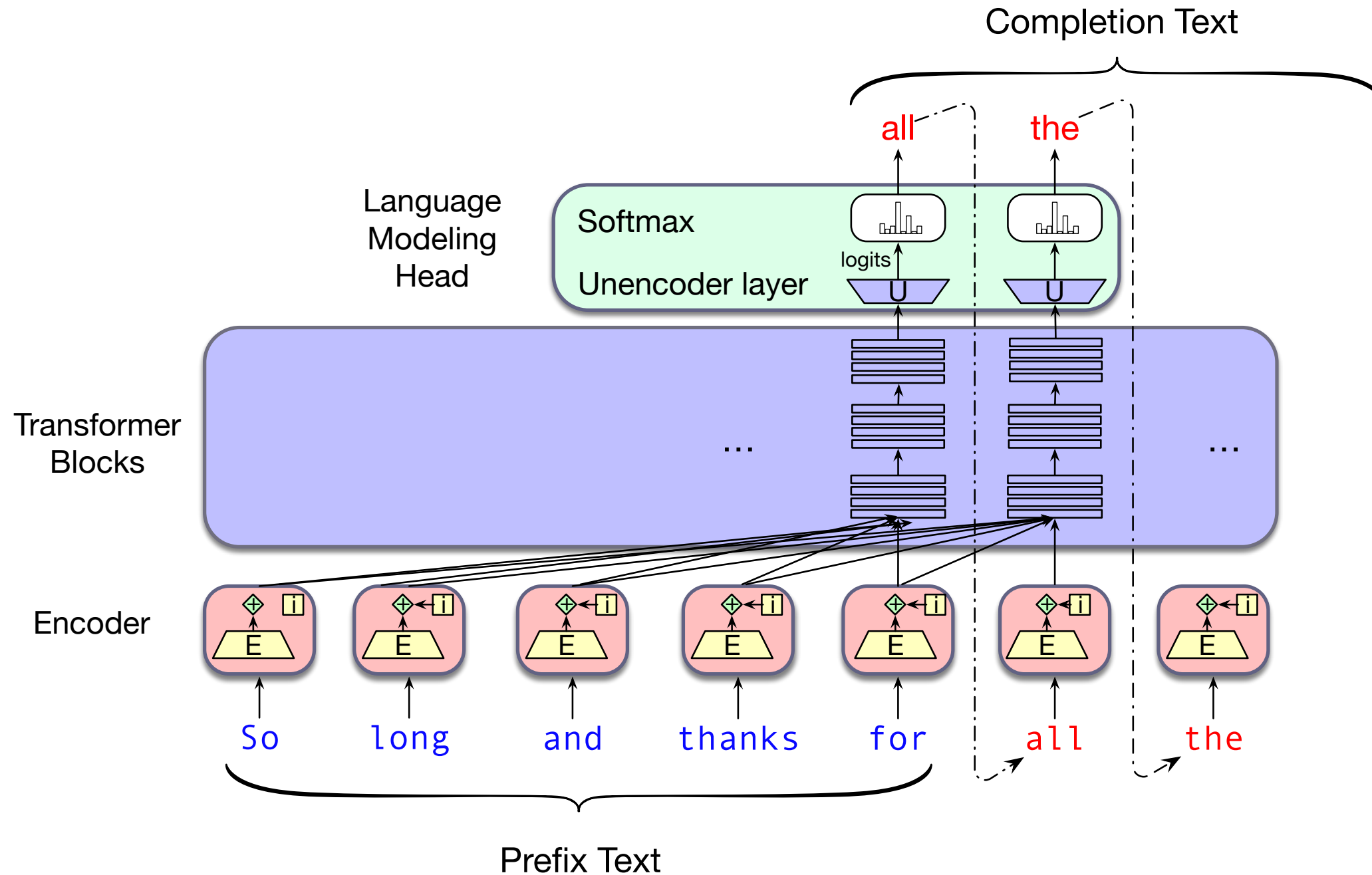


Words and Tokens

- Sentences are broken up into tokens, which roughly correspond to words or parts of words.
- Each token is represented as a vector (based on an initial learning phase), where similar tokens correspond to vectors that are close to each other.

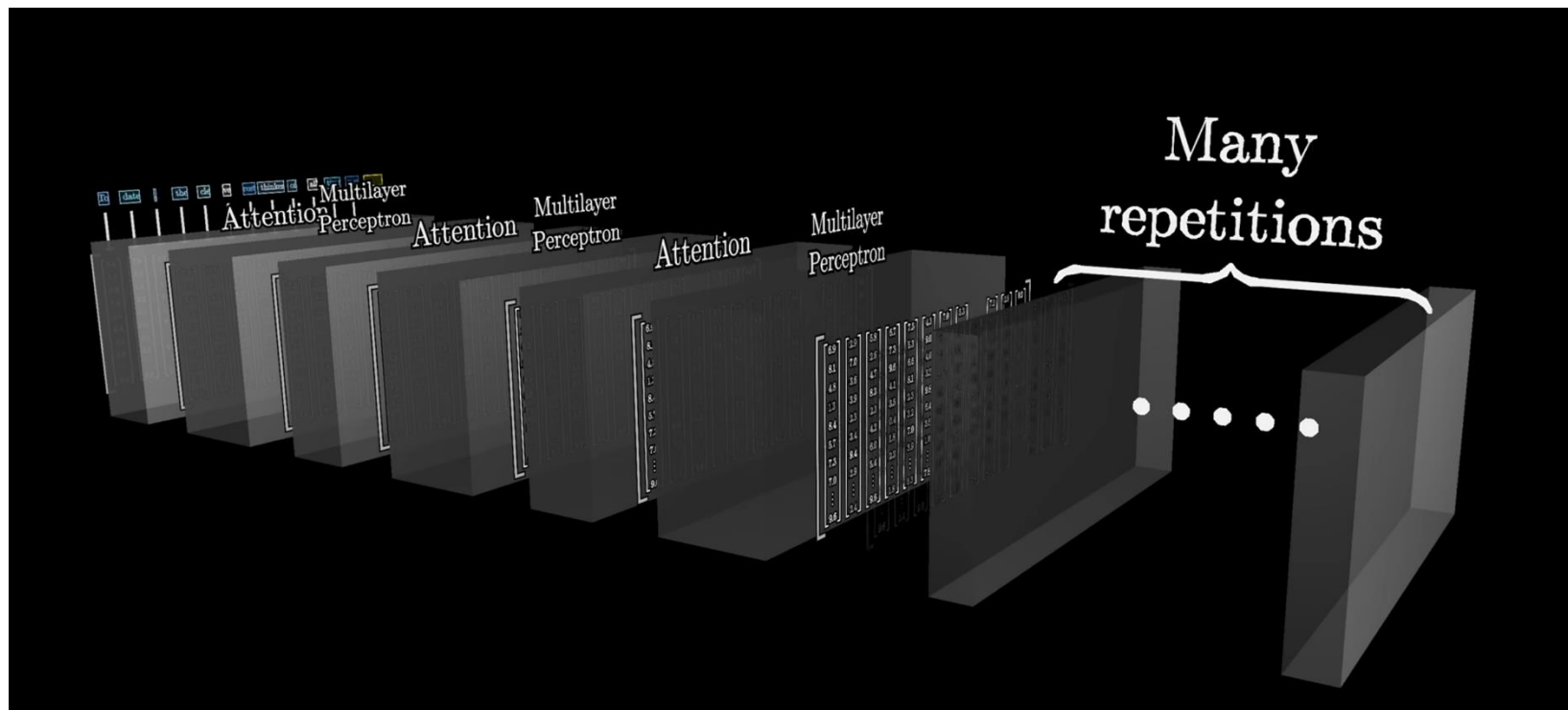


Conditional Generation: Generating text conditioned on previous text!



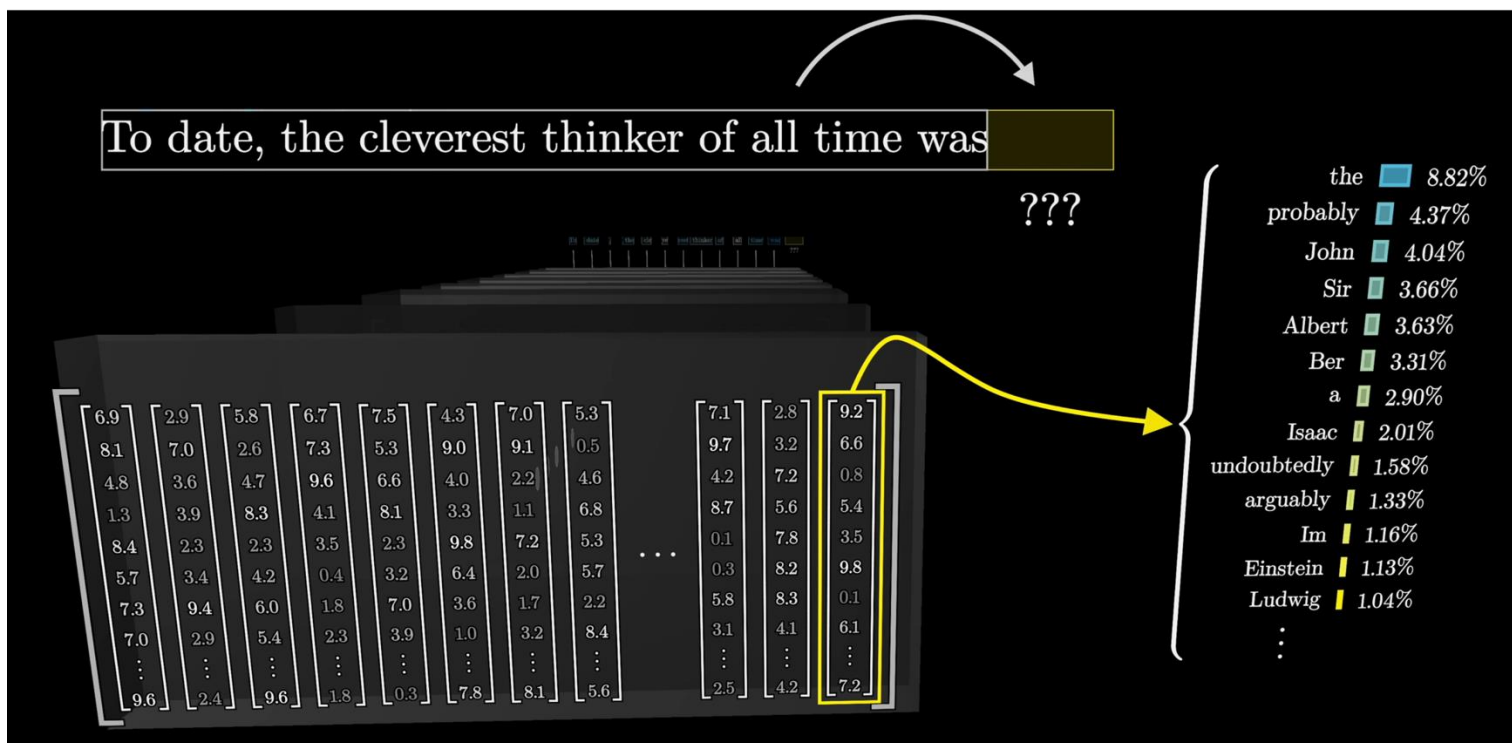
Layers in an LLM

- The model processes the vectors as matrices that are meant to encode semantic meaning (including word order and placement).



The Last Vector

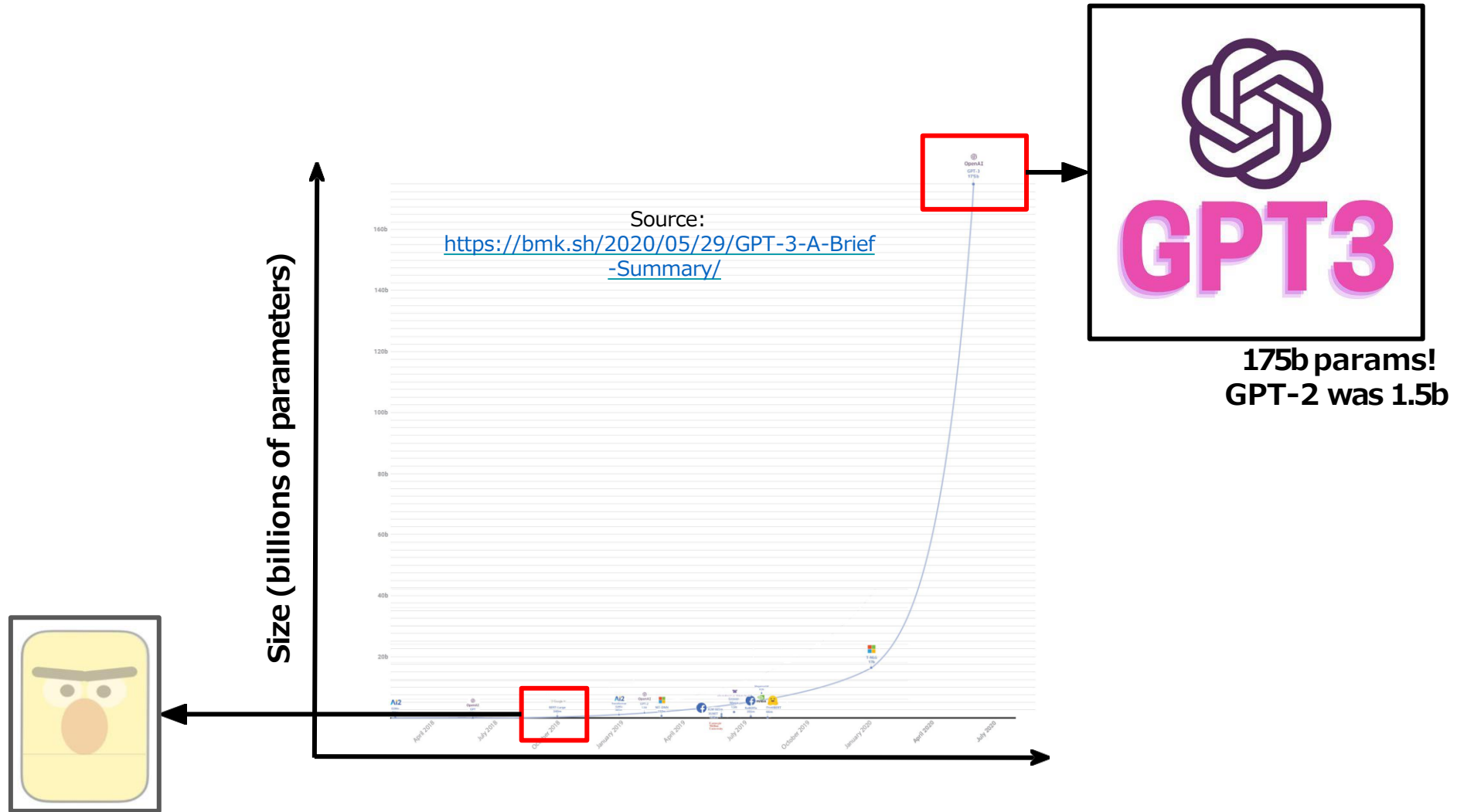
- The assumption is that the last vector encodes the answer to “what comes next” and then this is turned into token probabilities via a softmax.



Why did the transformer make such a big difference for language modeling?

It allowed for faster learning of more model parameters on more data (by allowing parallel computation on GPUs!)

Model Scaling: GPT-3



LLM Inference: Prompting

- Prompts
 - Tell the model what to do in natural language
 - For example, generate a textual summary of this paragraph:
 - Can be as short or long as required
- Prompt Engineering
 - The task of identifying the correct prompt needed to perform a task
 - General rule of thumb be as specific and descriptive as possible
 - Can be manual or automatic (prefix-tuning, paraphrasing etc.)

Large Language Models

The transformer model allows fast parallel computations on many GPUs (**large amounts of compute**)

It allows training on **large amounts of data** (think the whole internet worth of text).

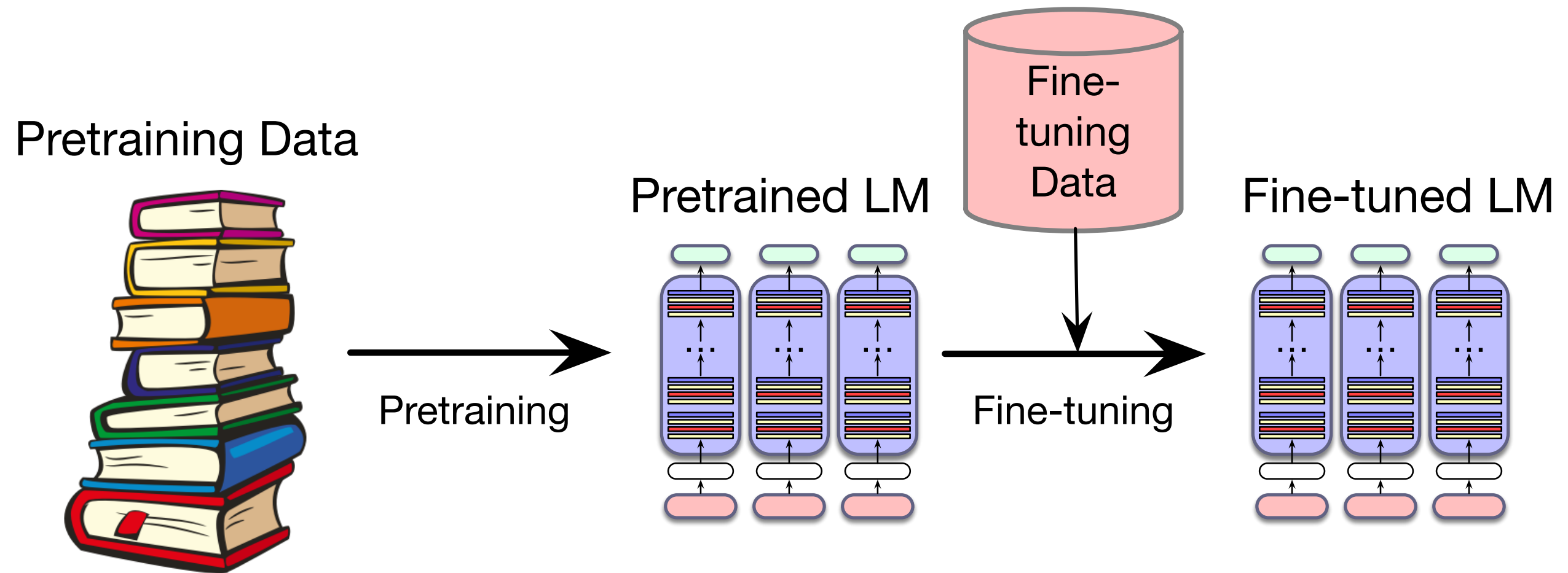
It allows adding many and many layers in the model (**large model**)

A large language model is a language model with a large number of parameters, trained on large amounts of data, for long period of time.

Why large language models?

- Scaling the models, compute, and data leads in increase in performance
- Emergent properties at scale (Wei et al 2022)
 - Large models (with 7-100B+ parameters) suddenly become capable of performing tasks they weren't able to do when small (such as 1B or small).

Finetuning



Training the model to chat

A simple language model (also called a pretrained model) is not equipped to chat with an end user, like ChatGPT.

ChatGPT (and many other models) are further trained on supervised data to follow instructions.

Instruction Tuning

- Collect a large dataset of instruction following examples of the form
 - <instruction> <input> <output>
 - For example,
 - Summarize this news article [ARTICLE] [SUMMARY]
 - Answer this question [QUESTION] [ANSWER]
 - Predict the sentiment of this review [REVIEW] [SENTIMENT]....
- This is also a text corpus but in a very specific format.
- Continue training the model on this dataset (again using the same training objective)

Aligning the model to humans' preferences

- Chat based on models are supposed to converse with humans
- Why not learn from humans' feedback
- Basic idea: Model samples multiple outputs – users rank them based on their preference
 - Convert user preferences into reward scores – more preferred output has higher reward
 - Treat an LLM like an agent and use RL to maximize this reward (RLHF)

So what does this mean ChatGPT is good at?

Some aspects of producing answers that might fall under that category:

- Writing in specific styles (that have appeared in the model's training data)
- Grammatical consistency
- Generating boilerplate sentences that often appear at the beginning, end of emails, etc.
- Fluency

What are some problems that ChatGPT's training leaves it prone to?

Inaccuracies

- The language model doesn't "plan" what it will say in advance
- The model doesn't store facts, just outputs plausible looking sentences which may or may not be factual

Hallucination

*Chatbots May 'Hallucinate'
More Often Than Many Realize*

What Can You Do When A.I. Lies About You?

People have little protection or recourse when the technology creates and spreads falsehoods about them.

Air Canada loses court case after its chatbot hallucinated fake policies to a customer

The airline argued that the chatbot itself was liable. The court disagreed.

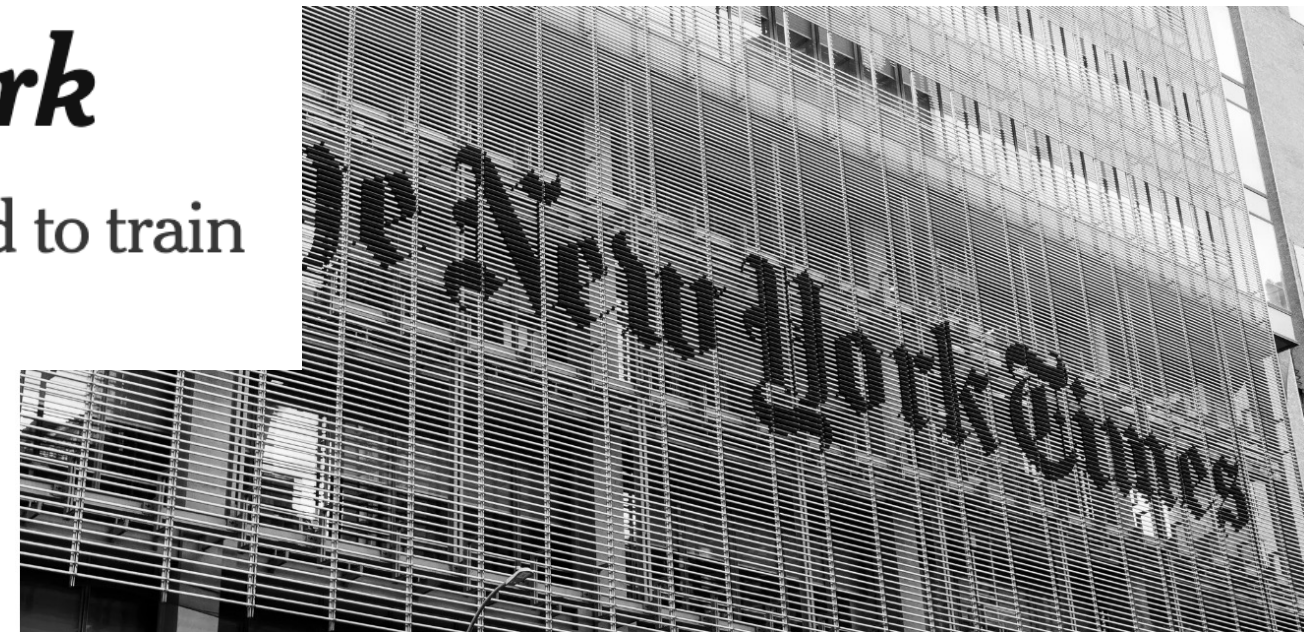
Copyright



Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.



Toxicity and Abuse

The New AI-Powered Bing Is Threatening Users.

Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Lack of source attribution

Just like the model doesn't store facts... it doesn't store sources.



Sure, here are the links to the papers I mentioned:



- Hazan and Kale (2016): Sublinear algorithms for approximating zero-sum games. <https://arxiv.org/abs/1606.05820>
- Chen, Gupta, and Roughgarden (2018): Faster algorithms for computing approximate Nash equilibria in bimatrix games. <https://arxiv.org/abs/1802.04974>
- Kamma, Kalai, and Tulsiani (2020): Entropic Mirror Descent for Solving Linear and Convex Games in Sublinear Time. <https://arxiv.org/abs/2007.05380>

I hope you find them helpful!

Outputs that reflect social biases

An example from machine translation a few years ago:

