

Introduction to Large Language Models (LLMs)

Murari Mandal
Assistant Professor

<https://murarimandal.github.io/>
<https://respailab.github.io/>

Course Instructors



Murari Mandal



Debdeep Sanyal



Aakash Sen Sharma



Umakanta Maharana



Arka Mukherjee

Website: <https://respailab.github.io/llm-101.respailab.github.io/>
RespAI Lab, KIIT Bhubaneswar

Acknowledgements

- Speech and Language Processing, Jurafsky & Martin
- Large Language Models (LLMs): Introduction and Recent Advances,
<https://lcs2-iitd.github.io/ELL881-AIL821-2401/>
- Hands-on Large Language Models, Jay Alammar & Maarten Grootendorst
- Building Large Language Model (from scratch), Sebastian Raschka
- <https://www.youtube.com/andrejkarpathy>, Andrej Karpathy

Importance of Language?

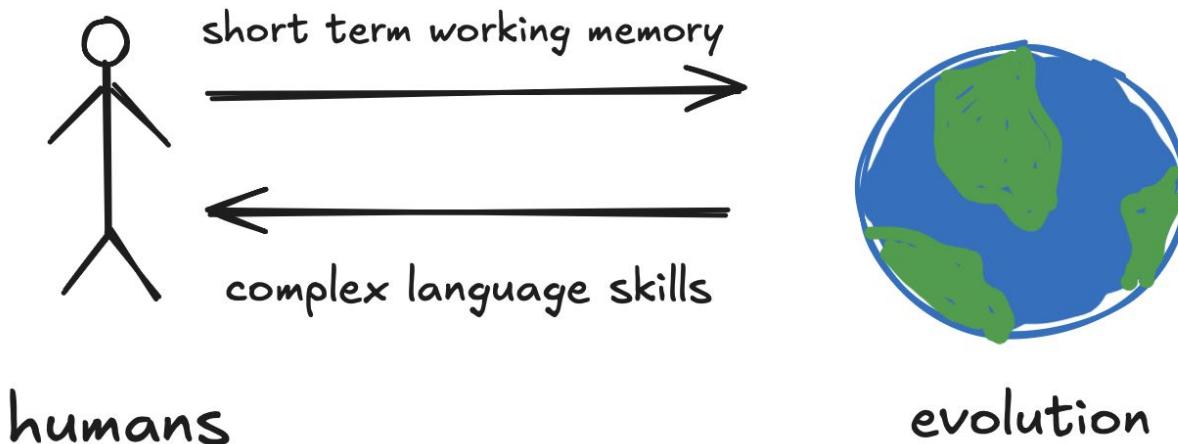


Chimpanzees can *memorize* patterns of numbers at a glance, something which is beyond humans

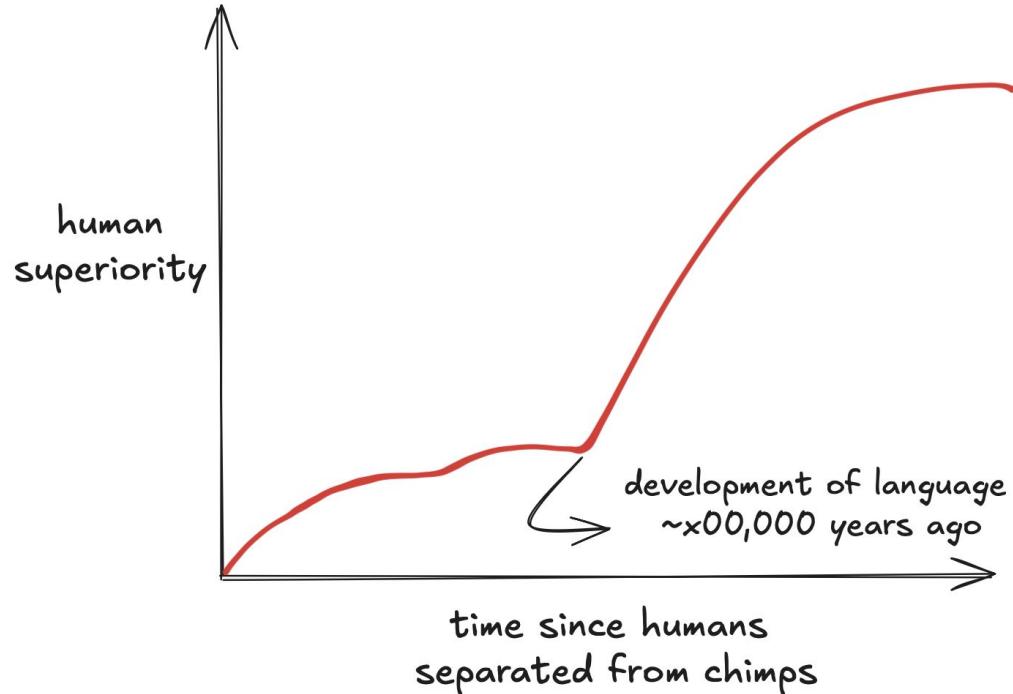
[https://www.cell.com/current-biology/pdf/S0960-9822\(07\)02088-X.pdf](https://www.cell.com/current-biology/pdf/S0960-9822(07)02088-X.pdf)

The Cognitive Trade-off Hypothesis

- Human brain has limited resources, improvements in one *cognitive area* may come at the expense of another.
- A trade-off between *superior language facility* at the expense of *memory ability*



The Cognitive Trade-off Hypothesis



Language Model

Computes a *probability* for each word in a sequence, determining how *likely* each word is given the *previous context*.

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?
What is $P(\text{text})$?

Vocabulary
 $V = \{\text{Jane, went, to, the, The, store, food, truck}\}$

Language Model

$P(\text{Jane went to the store})$ ► 3.05×10^{-5}
$P(\text{store to Jane went the})$ ► 3.05×10^{-5}
$P(\text{Jane went store})$ ► 1.9×10^{-3}
$P(\text{Jane goed to the store})$ ► 0.0
$P(\text{The store went to Jane})$ ► 3.05×10^{-5}
$P(\text{The food truck went to Jane})$ ► 3.8×10^{-6}

Language
Model

Vocabulary

$V = \{\text{Jane, went, to, the, The, store, food, truck}\}$

Language Model

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

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

Language Model

What is a Language Model?

A language model answers the question: **What is P(text)?**

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

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

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

Using the chain
rule of probability

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

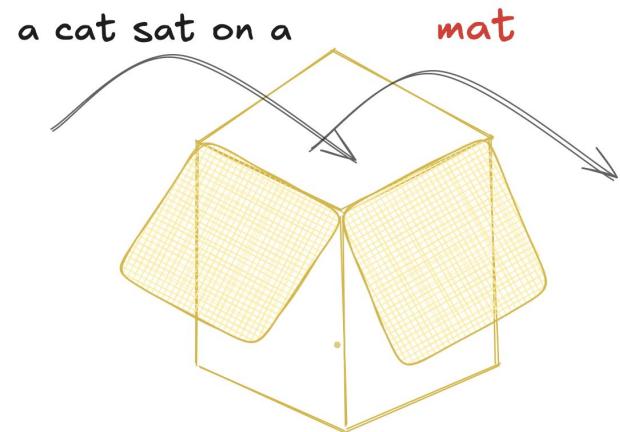
Language Model

What is a Language Model?

$$P(w_n | w_{1:n})$$

given this sequence
of words from 1 to n

what is the probability
of generating this
current word n



Language Model

LM deals in *tokens*. So a LM gives the probability distribution over a sequence of tokens.

"Tokenization example"

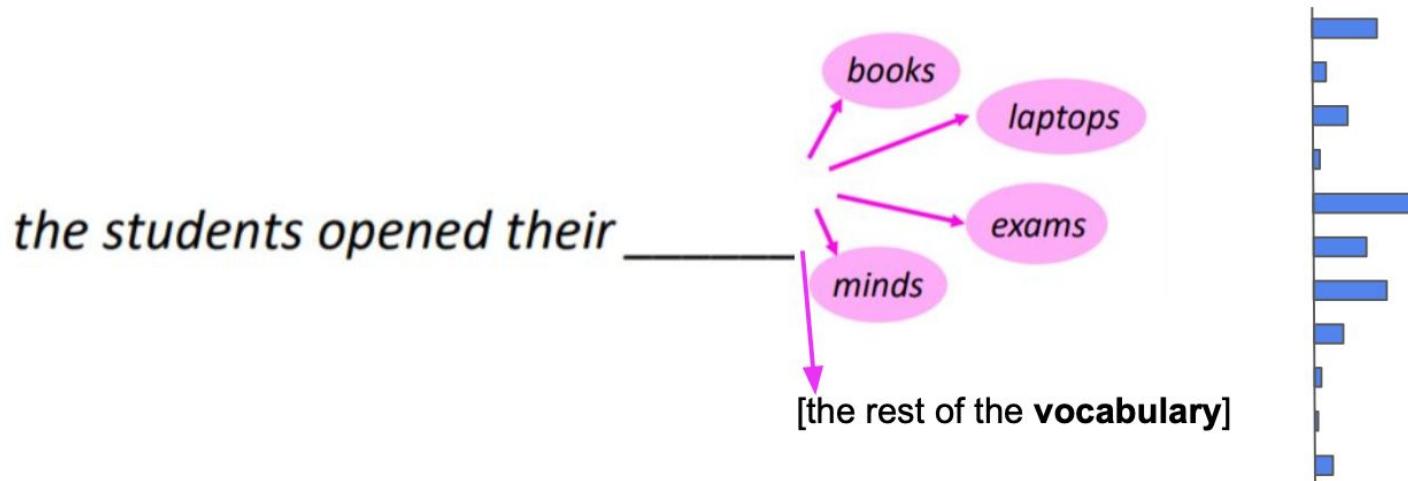
Word-Level Tokenization: ["Tokenization", "example"]

Subword Tokenization: ["Token", "ization", "example"]

More on this later...

Language Model

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



Language Model

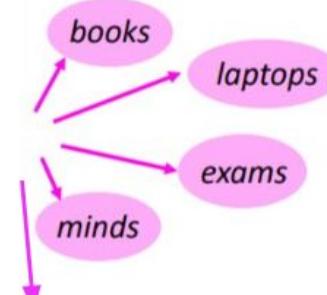
At each timestep, sample a token from the LM's new probability distribution over next tokens.

The _____

The students _____

The students opened _____

The students opened their _____

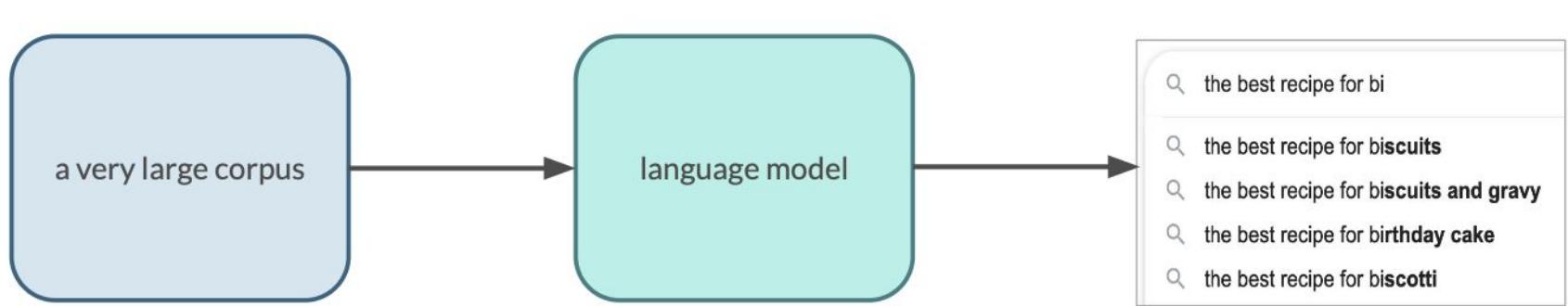


[the rest of the LM's vocabulary]

In short, **generating** next token

Language Model

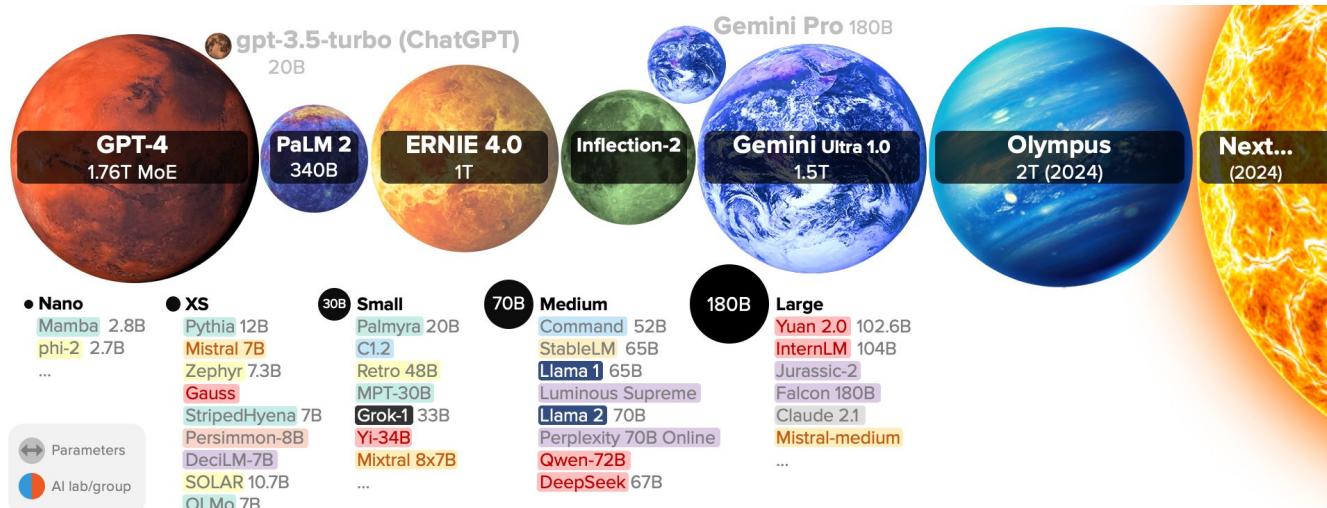
LM can generate text



What is a ‘Large’ Language Model (LLM) ?

The ‘Large’ in LLM implies

- Large model size (**# parameters**)
- Large size of training dataset (**# tokens**)



Disclaimer: Closed models such as GPT-4 and Gemini-Ultra, the official parameter counts have not been disclosed—these figures are based on online rumors.



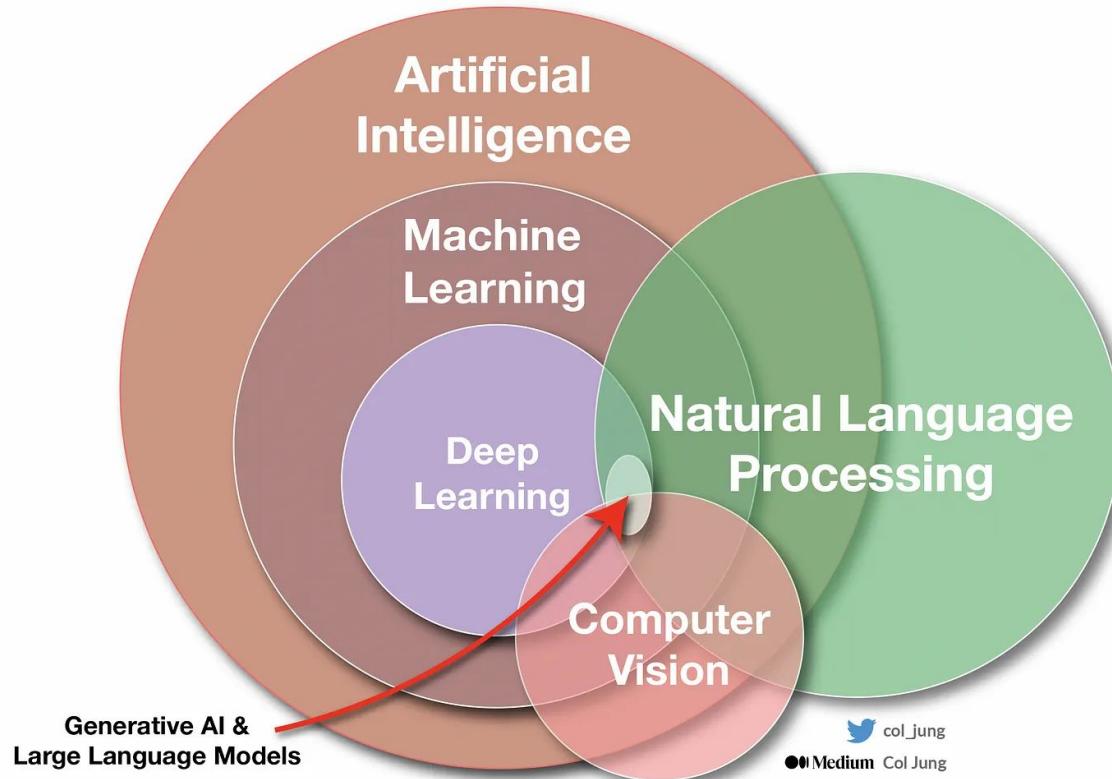
LifeArchitect.ai/models

What is a ‘Large’ Language Model ?

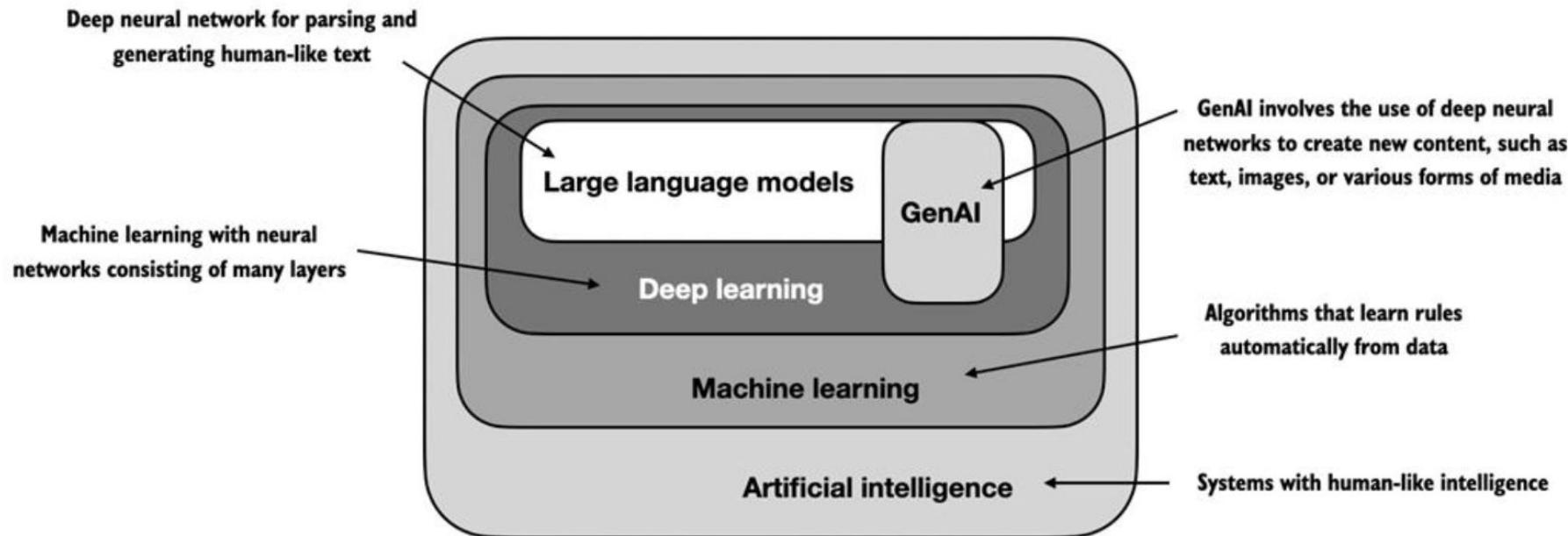
Corpora	Size	Source	Latest Update Time
BookCorpus [158]	5GB	Books	Dec-2015
Gutenberg [159]	-	Books	Dec-2021
C4 [82]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [160]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWS [161]	120GB	CommonCrawl	Apr-2019
OpenWebText [162]	38GB	Reddit links	Mar-2023
Pushift.io [163]	2TB	Reddit links	Mar-2023
Wikipedia [164]	21GB	Wikipedia	Mar-2023
BigQuery [165]	-	Codes	Mar-2023
the Pile [166]	800GB	Other	Dec-2020
ROOTS [167]	1.6TB	Other	Jun-2022

Company	Model	Size (B)	Data Scale	Open Resource	Release Time
Meta	LLaMA [65]	7B,13B,30B,65B	1.4T tokens	✓	Feb-2023
	LLaMA2 [66]	7B,13B,70B	2T tokens	✓	Jul-2023
	LLaMA3 [25]	8B,70B,405B	15T tokens	✓	Apr-2024
DeepSeek	DeepSeek-V1 [67]	7B,67B	2T tokens	✓	Dec-2023
	DeepSeek-V2 [58]	236B	8.1T tokens	✓	May-2024
	DeepSeek-V3 [68]	671B	14.8T tokens	✓	Dec-2024
	DeepSeek-R1 [28]	671B	-	✓	Jan-2025
Qwen	Qwen1.5 [69]	0.5B,1.8B,4B,7B,14B,72B	3T tokens	✓	Sep-2023
	Qwen2 [70]	0.5B,1.5B,7B,57B,72B	7T tokens	✓	Jul-2024
	Qwen2.5 [71]	0.5B,1.5B,3B,7B,14B,72B	18T tokens	✓	Dec-2024
Mistral	Mistral-7B [72]	7B	-	✓	Sep-2023
	Mistral-8X7B [55]	4.7B	-	✓	Dec-2023
	Mistral-8X22B [55]	14B	-	✓	Apr-2024
	Mistral-Large2 [73]	12.3B	-	✓	Jul-2024
Anthropic	Claude2 [74]	2B	-	✗	Jul-2023
	Claude3 [27]	-	-	✗	Mar-2024
	Claude3.5 [75]	175B	-	✗	Oct-2024
Google	Gemini1.0 [76]	-	-	✗	Dec-2023
	Gemini1.5 [77]	-	-	✗	Mar-2024
	Gemini2.0 [26]	-	-	✗	Dec-2024
OpenAI	GPT-1 [1]	0.01B	-	✗	Jun-2018
	GPT-2 [14]	0.15B	-	✗	Feb-2019
	GPT-3 [7]	17.5B	-	✗	Jul-2020
	InstructGPT [45]	-	-	✗	Mar-2022
	GPT-4 [9]	180B	-	✗	Mar-2023
	GPT-4o [78]	200B	-	✗	Oct-2024
	o1 [41]	300B	-	✗	Dec-2024
	o3-mini [79]	-	-	✗	Jan-2025
ZhiPuAI	GLM-4[80]	130B	-	✓	Jun-2024
Databricks	DBRX [81]	132B	-	✓	Mar-2024
01.AI	Yi-Large [82]	-	-	✓	Mar-2024
AI21 Labs	Jamba1.5 Large [83]	94B	-	✓	Aug-2024
Amazon	Nova Pro [84]	-	-	✗	Dec-2024
MoonshotAI	Kimi-k1.5 [85]	-	-	✗	Jan-2025

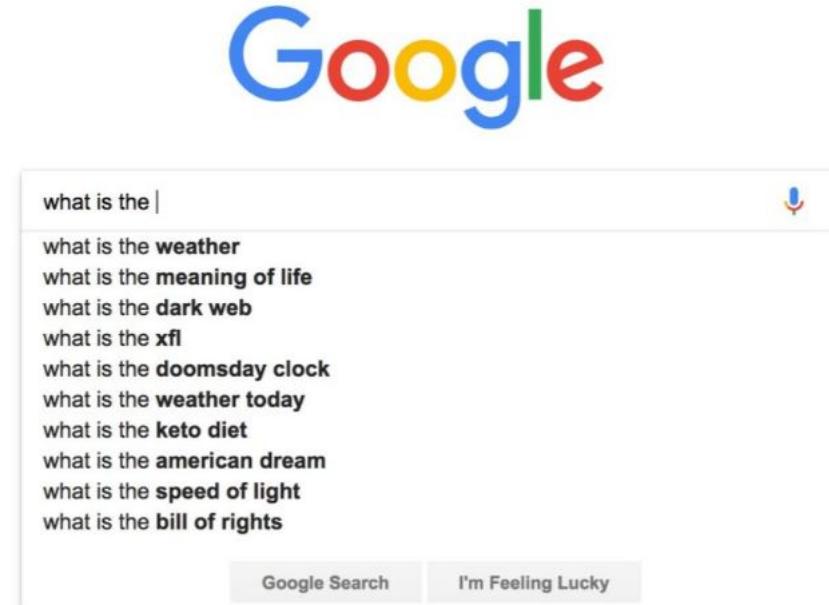
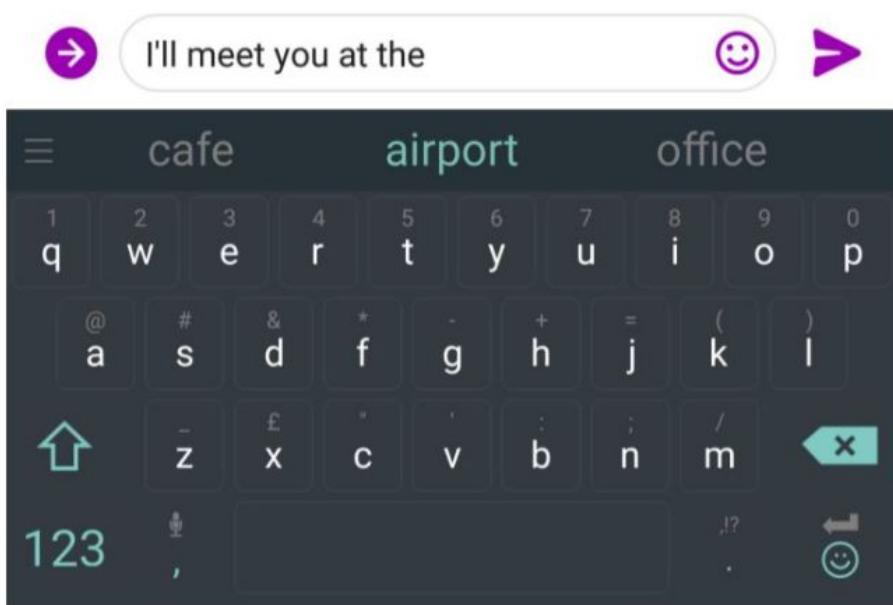
What is a ‘Large’ Language Model ?



What is a ‘Large’ Language Model ?



Where do we use LM?



Where do we use LM?

utils.js - Untitled (Workspace) - Visual Studio Code - Insiders

it Selection View Go Run Terminal Help

GITHUB COPilot

amitmercha1990 write tests for this code

GitHub Copilot

Sure, here are some test cases you can use to test the calculateCharactersAndWords function:

```
describe('calculateCharactersAndWords', () => {
  it('should return "0 characters, 0 words" for an empty string', () => {
    const result = calculateCharactersAndWords('');
    expect(result).toEqual('0 character(s), 0 word(s)');
  });

  it('should return the correct character and word count for a single word', () => {
    const result = calculateCharactersAndWords('Hello');
    expect(result).toEqual('5 character(s), 1 word(s)');
  });

  it('should return the correct character and word count for multiple words', () => {
    const result = calculateCharactersAndWords('Hello world');
    expect(result).toEqual('14 character(s), 2 word(s)');
  });
});

it('should return the correct character and word count for a sentence', () => {
  const result = calculateCharactersAndWords('Hello world! This is a test.');
  expect(result).toEqual('34 character(s), 6 word(s)');
});
```

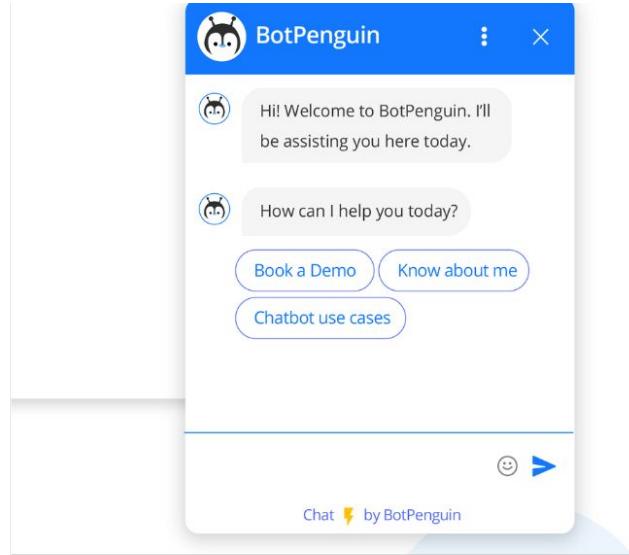
Ask Copilot a question or type '?' for topics

utils.js

```
reset() => // utils.js @ calculateCharactersAndWords
  102
  103
  104  function resetShowWordCountPill(defaultShowWordCount) {
  105    defaultShowWordCountPill === 'Yes' ? $(`#${showWordCountPill}`).val(defaultShowWordCount)
  106    107
  108
  109  function countWords(str) {
  110    return str.trim().split(/\s+/).length;
  111  }
  112  113  function calculateCharactersAndWords(str) {
  114    const characterCount = str.length;
  115    const wordCount = str.split(' ').length;
  116    const wordCountText = `${characterCount} character(s), ${wordCount} word(s)`;
  117
  118    return wordCountText;
  119  }
  120
  121  function copyNotesToClipboard(note) {
  122    navigator.clipboard.writeText(note).then(function() {
  123      showToast('Notes copied to clipboard!');
  124    }, function() {
  125      showToast('Failure to copy. Check permissions');
  126    });
  127  }
  128}
```

Ln 113 Col 1 (53 selected) Spaces: 4 UFT-8 ⌂ (.) JavaScript

Code generation



Chatbots



Translators

Where do we use LM?

Amazon uses LLMs to have "common sense reasoning" with users buying products.



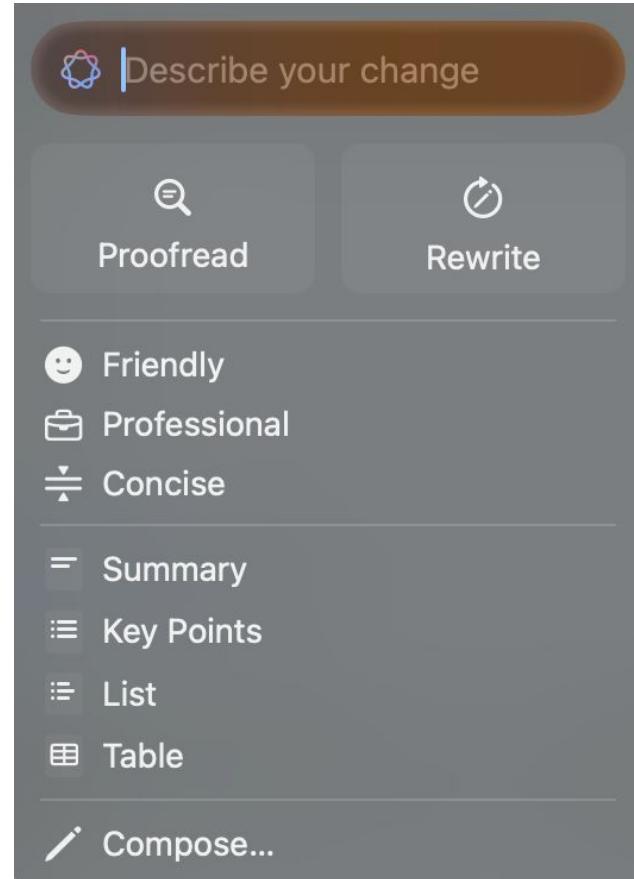
Where do we use LM?

Code autogeneration
with Github CoPilot,
Cursor, Gemini Code Review

```
▶ input = 'Write me a haiku'  
model_inputs = tokenizer(input, return_tensors='pt')  
outputs = model.generate(  
    **model_inputs.to("cuda"),  
    max_new_tokens = 64, # Increase for longer outputs!  
    # Recommended Gemma-3 settings!  
    temperature = 1.0, top_p = 0.9
```

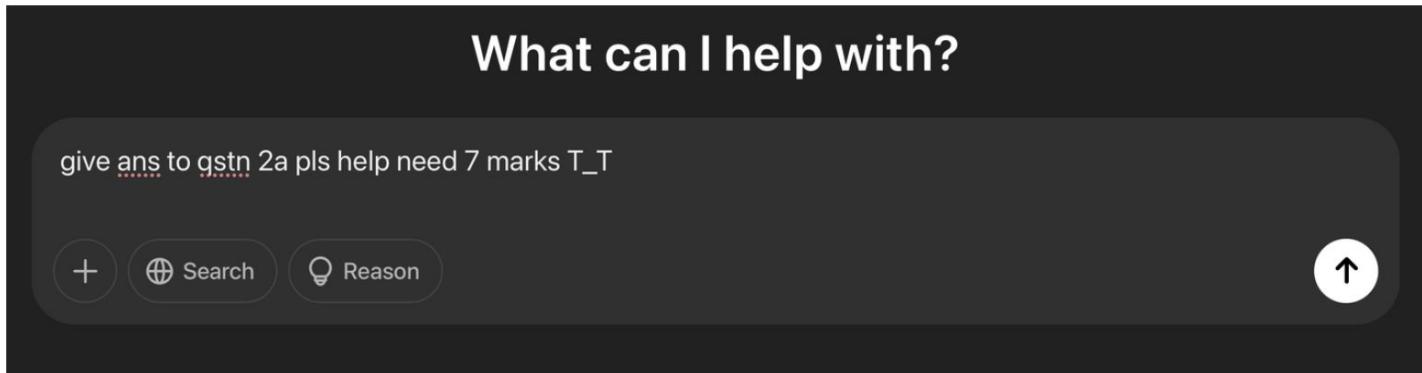
Where do we use LM?

Apple has a writing assistant
which can change your text
however you wish to.

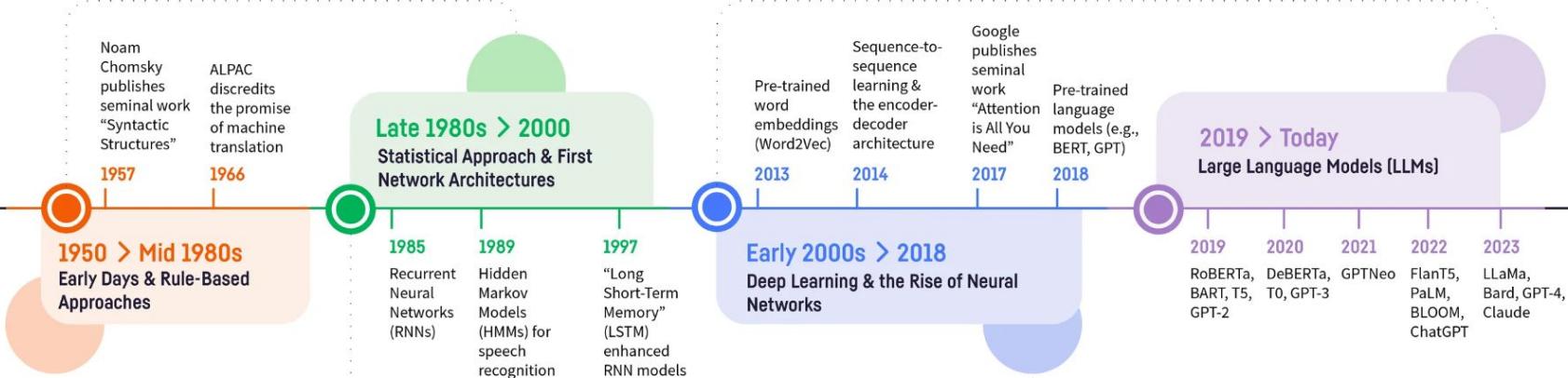


Where do we use LM? Many more Applications of LMs...

Basically, LLMs are a lot more useful than this.



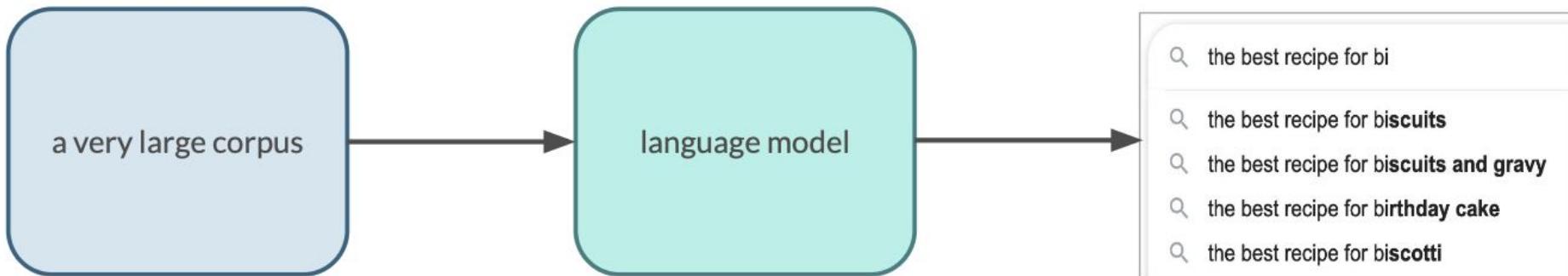
Evolution of (L)LMs



Building LMs

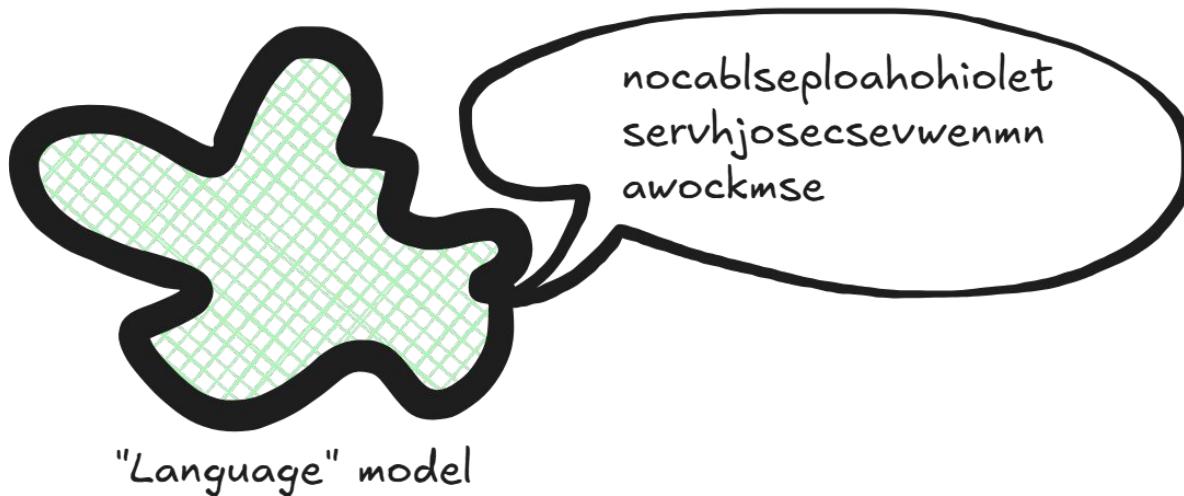
Estimate probabilities using text data

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

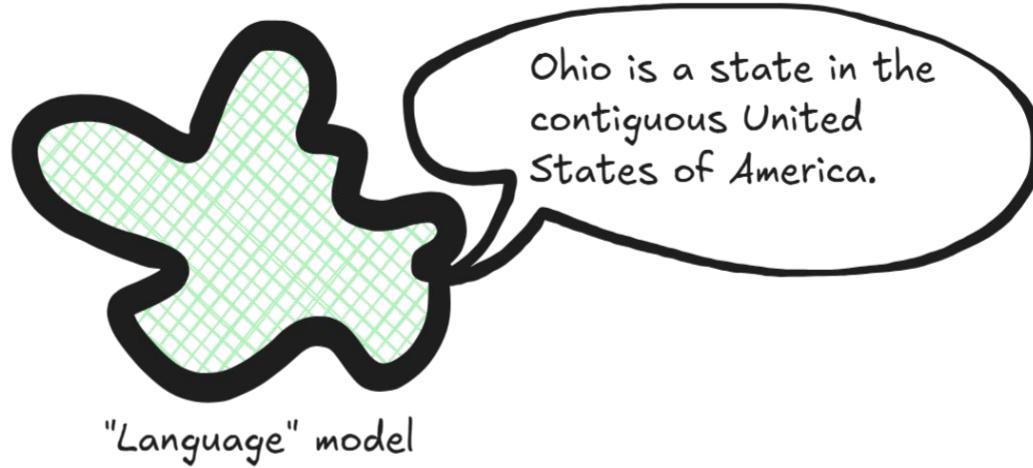


Building LMs

Naive idea: an algorithm that randomly generates a letter at a time



Building LMs

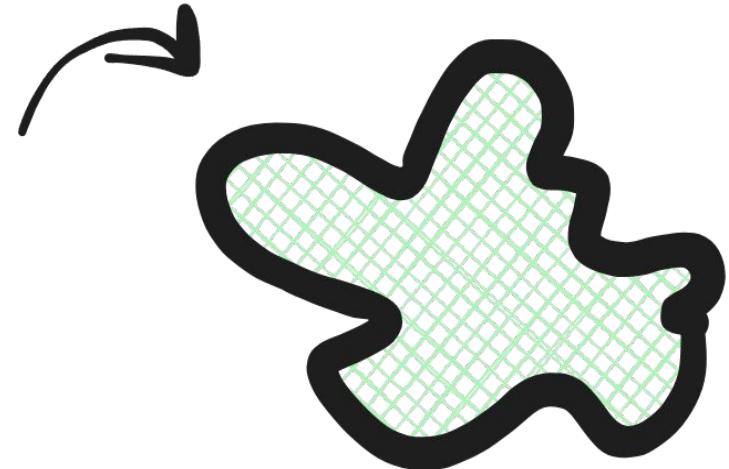


We need the model to generate coherent text

Building LMs: Rule-based Algos (1950s)

Hard-code the text to display

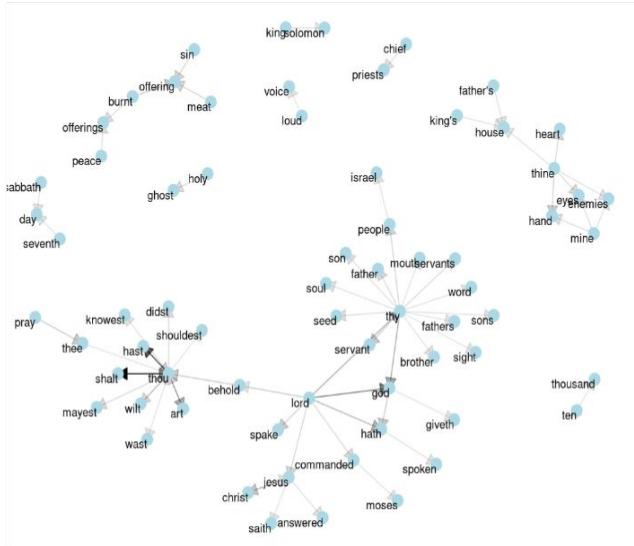
```
if 'Ohio' in prompt:  
    print("Ohio is a state  
in the contiguous United  
States of America.")  
else:  
    print("I don't know. I  
am just an if-else loop!..")
```



"Language" model

Building LMs: Statistical Methods (1970s)

Statistically determine the most likely word/letter that could follow



Markov chain of the Bible



n-gram language model

Building LMs: Statistical Methods (1970s)

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.
- The Language Model:

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

No generalization!

Building LMs: Statistical Methods (1970s)

Markov assumption

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

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

Building LMs: Statistical Methods (1970s)

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

or maybe even

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

Building LMs: Statistical Methods (1970s)

N-gram LM

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

"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

Building LMs: Statistical Methods (1970s)

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

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Building LMs: Statistical Methods (1970s)

Bigram Probability $P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$

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

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

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

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

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

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

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Building LMs: Statistical Methods (1970s)

Bigram Probability

Example:

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

< s > I am Sam </ s >
< s > Sam I am </ s >
< s > I do not like green eggs and ham </ s >

$$P(I | < s >) = \frac{2}{3} = .67 \quad P(Sam | < s >) = \frac{1}{3} = .33 \quad P(am | I) = \frac{2}{3} = .67$$

$$P(</ s > | Sam) = \frac{1}{2} = 0.5 \quad P(Sam | am) = \frac{1}{2} = .5 \quad P(do | I) = \frac{1}{3} = .33$$

Building LMs: Statistical Methods (1970s)

Trigram Probability

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

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

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

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

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

Building LMs: Statistical Methods (1970s)

n-gram Probability

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

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

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

Building LMs: Statistical Methods (1970s)

Sampling from n-gram LMs

1
gram

- To him swallowed confess hear both. Which. Of save on trail for are ay device and
rote life have
- Hill he late speaks; or! a more to leg less first you enter

2
gram

- Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live
king. Follow.
- What means, sir. I confess she? then all sorts, he is trim, captain.

3
gram

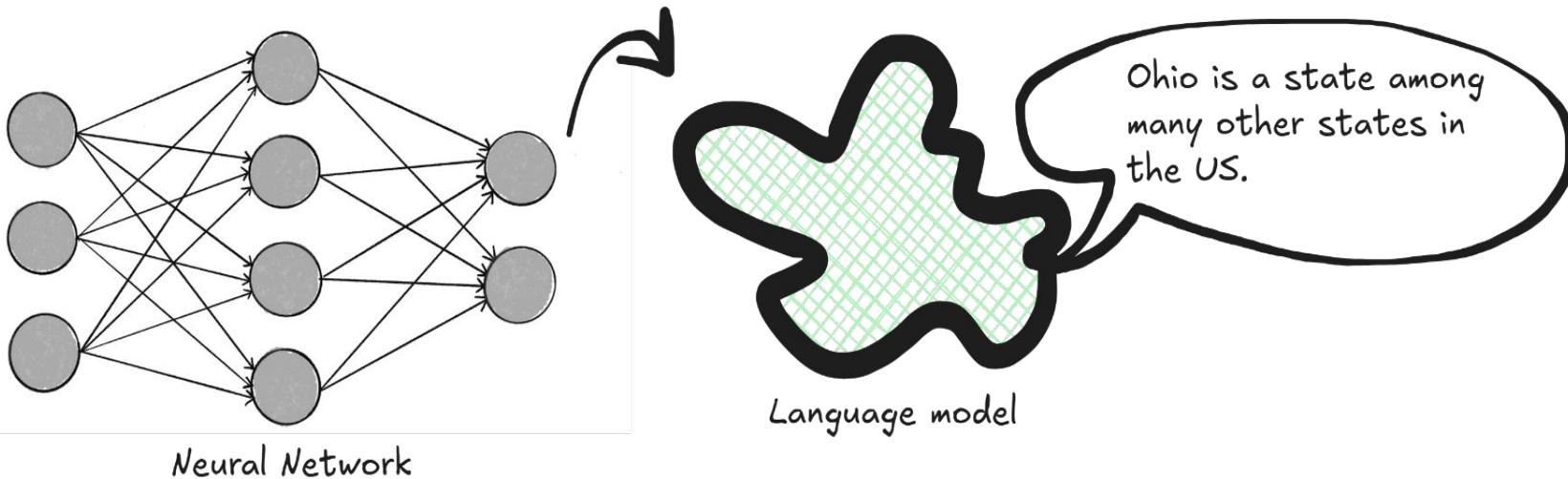
- Fly, and will rid me these news of price. Therefore the sadness of parting, as they say,
'tis done.
- This shall forbid it should be branded, if renown made it empty.

4
gram

- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A
great banquet serv'd in;
- It cannot be but so.

Building LMs: Neural Networks (1990s-2017)

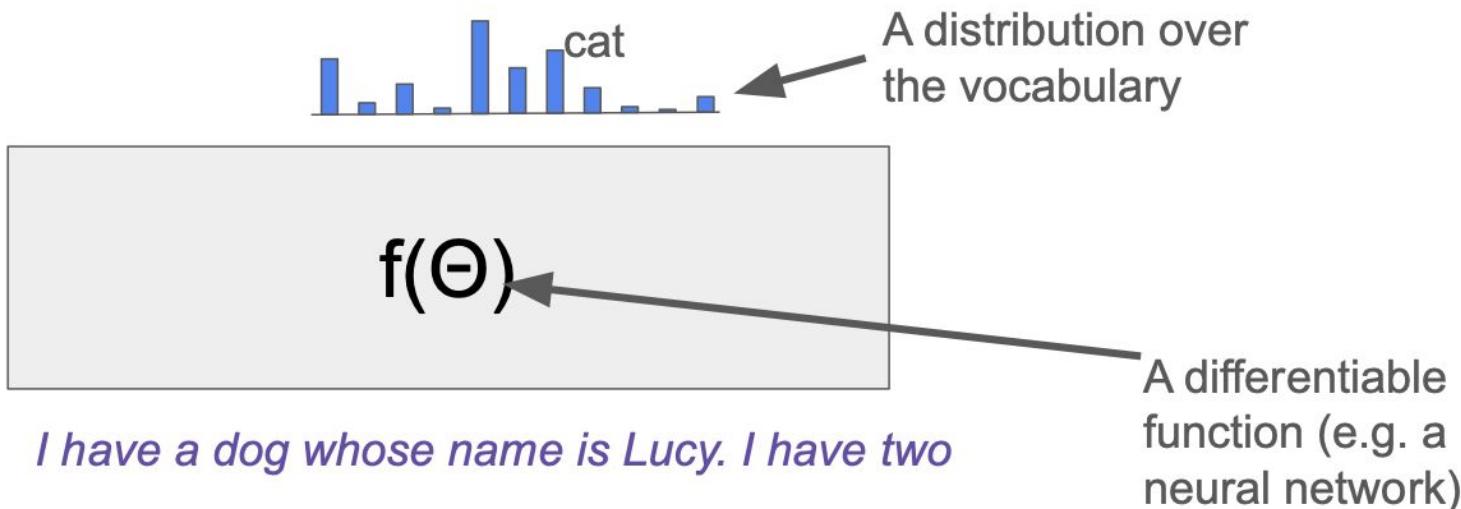
Train complex neural networks to intelligently predict the next word.



Building LMs: Neural Networks (1990s-2017)

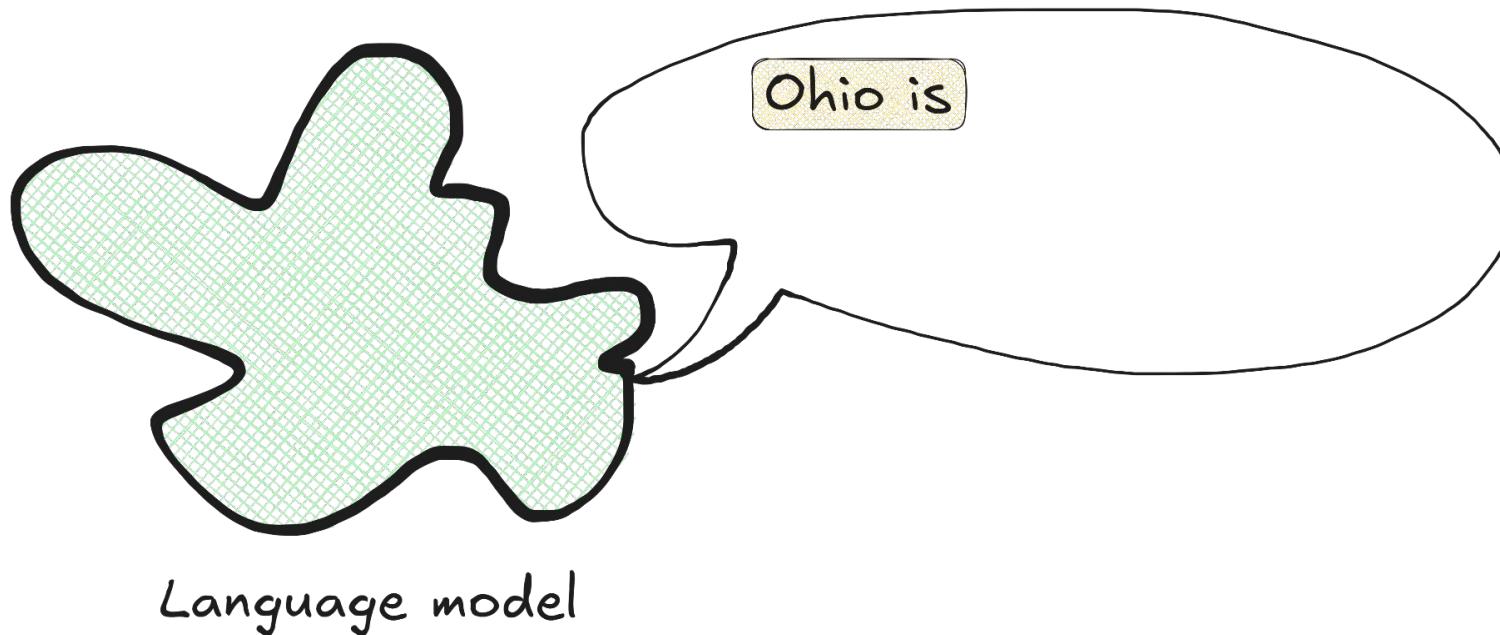
Train complex neural networks to intelligently predict the next word.

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



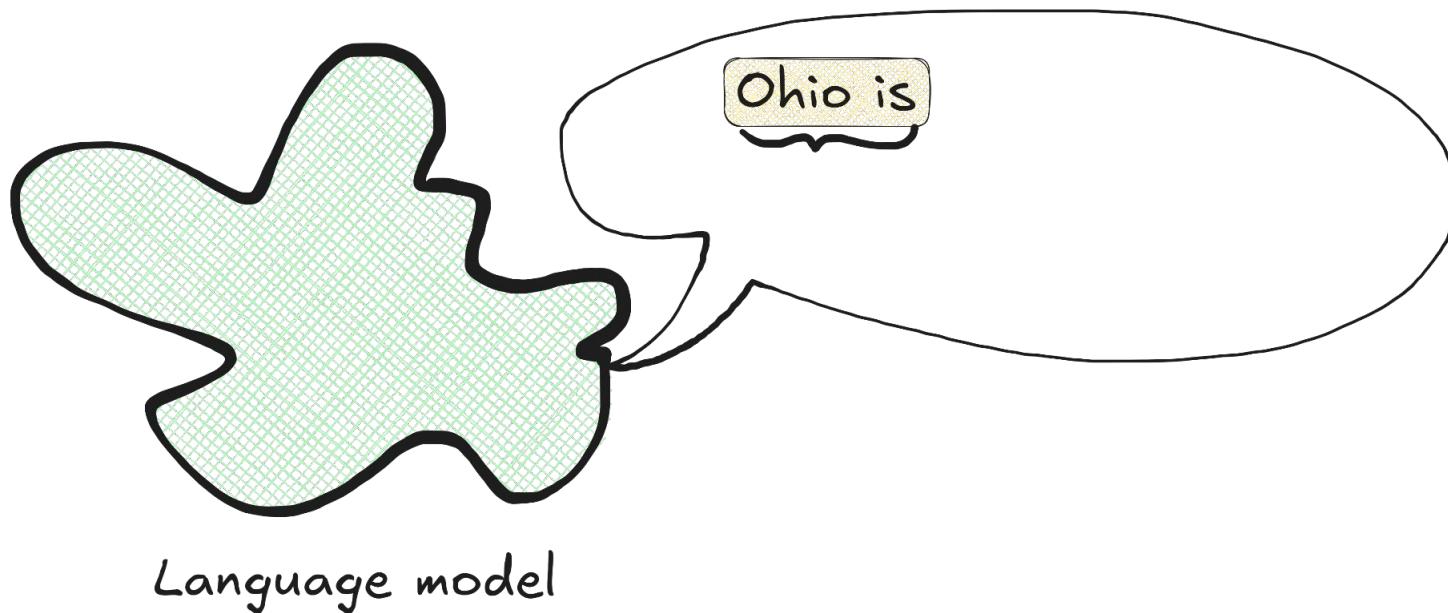
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



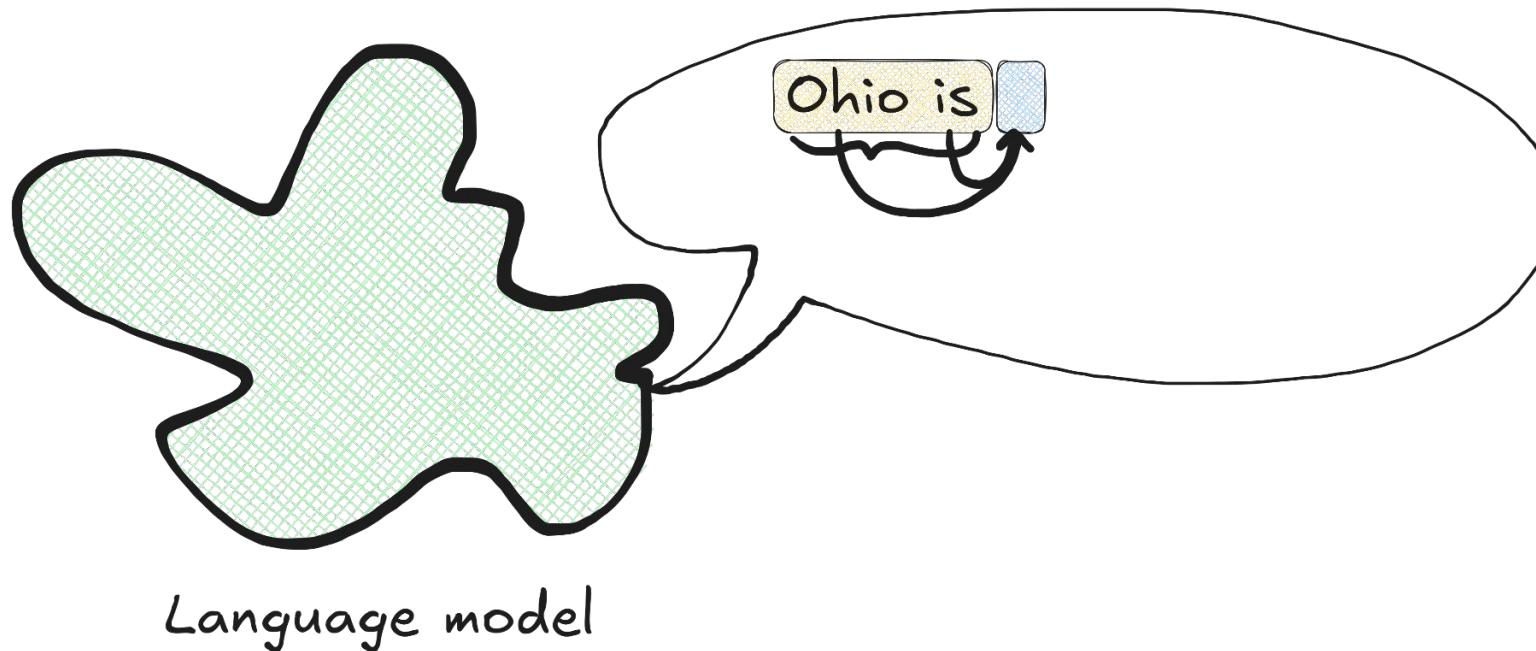
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



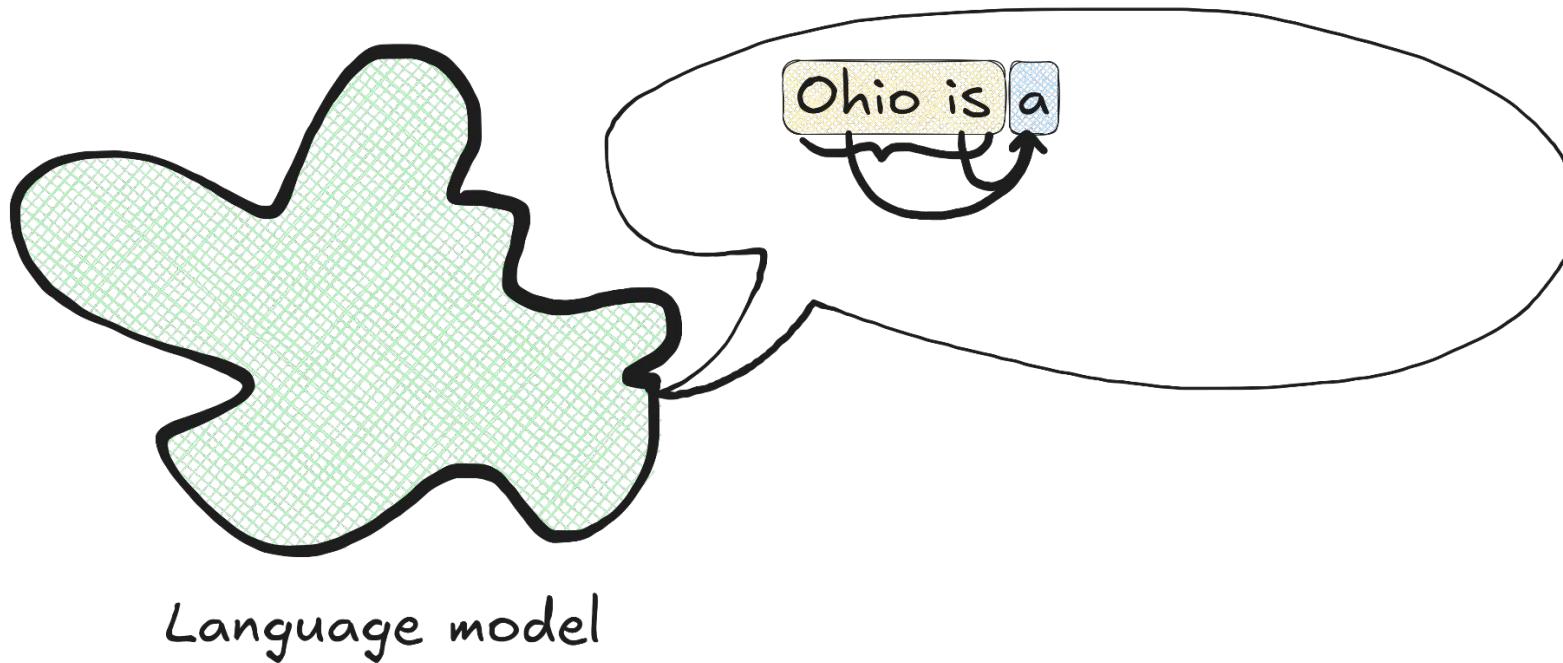
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



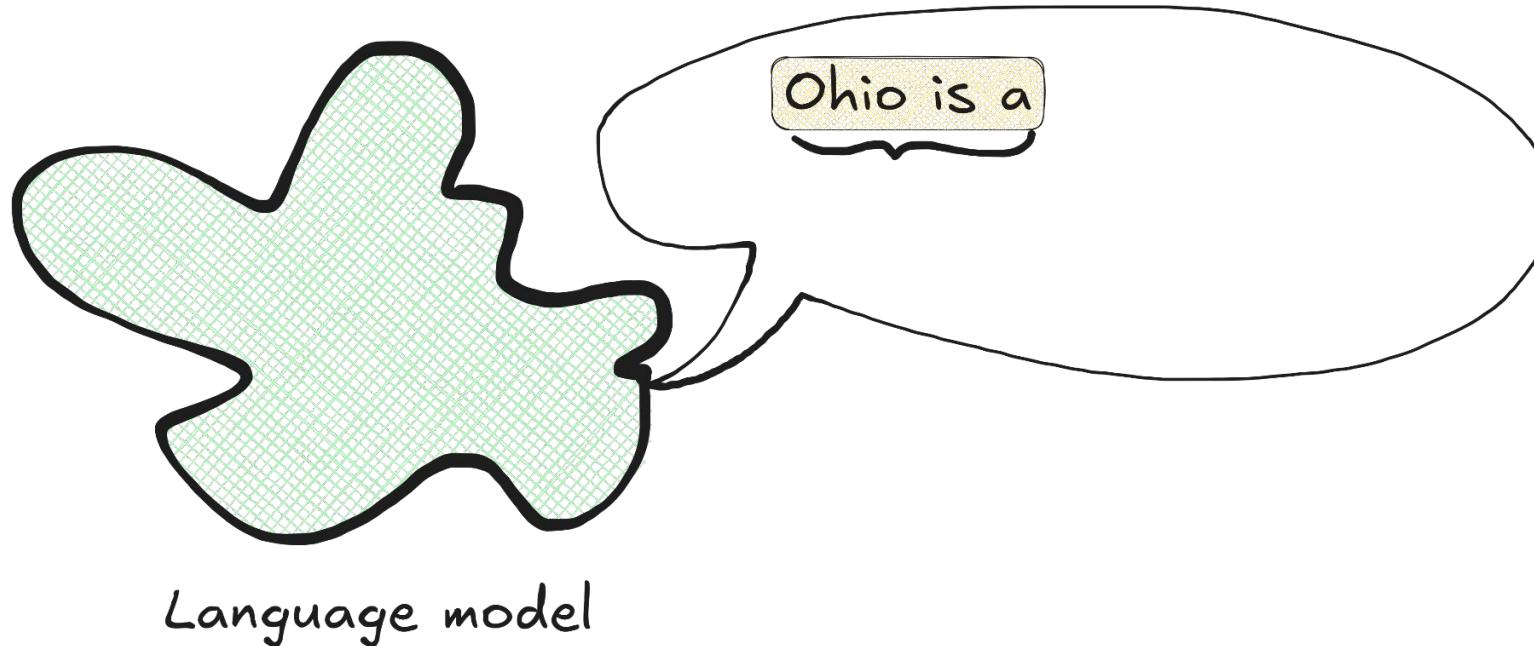
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



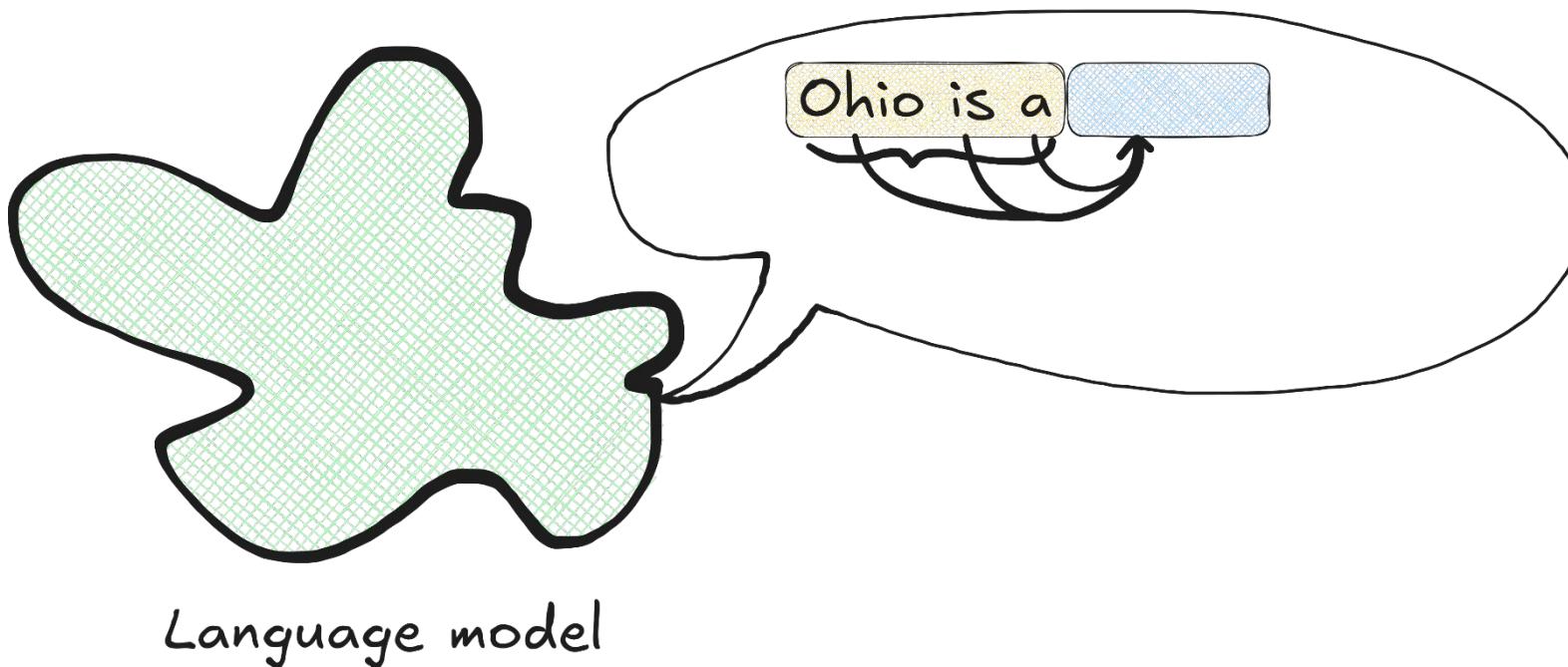
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



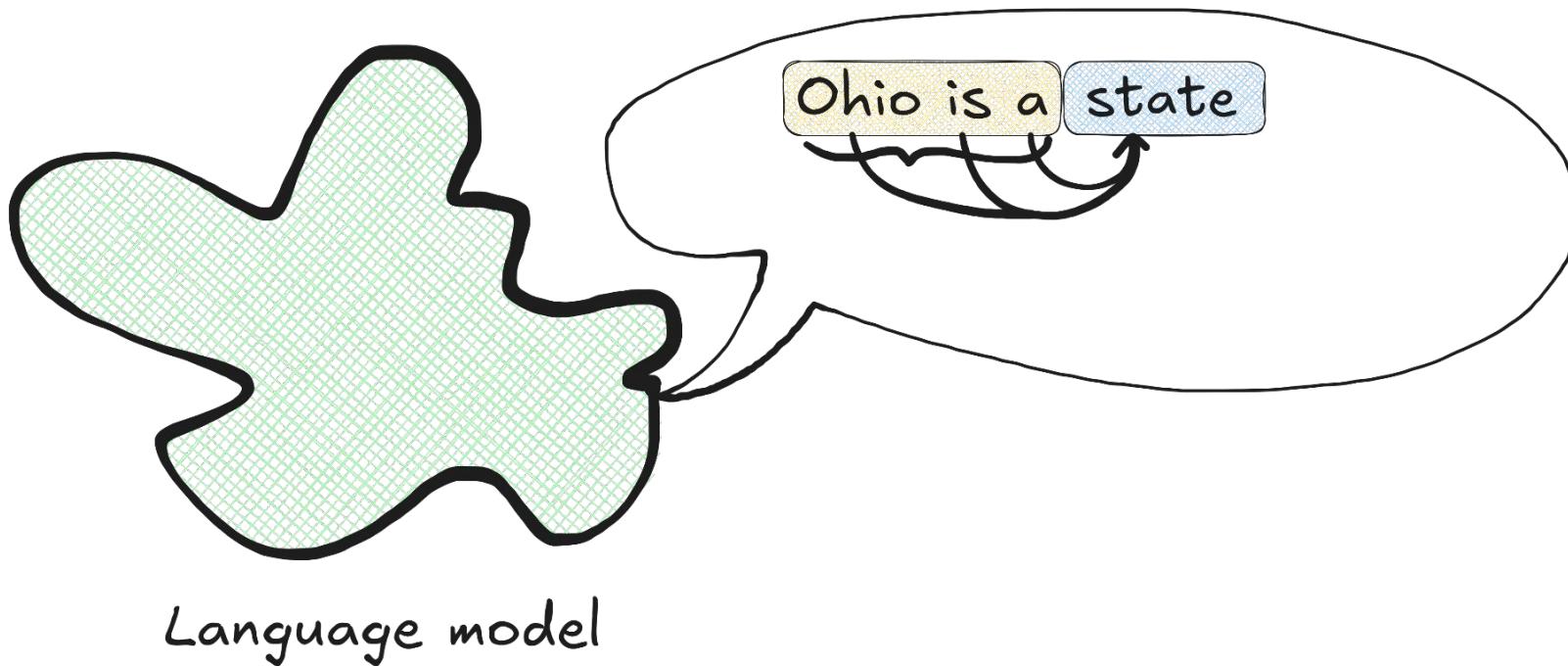
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



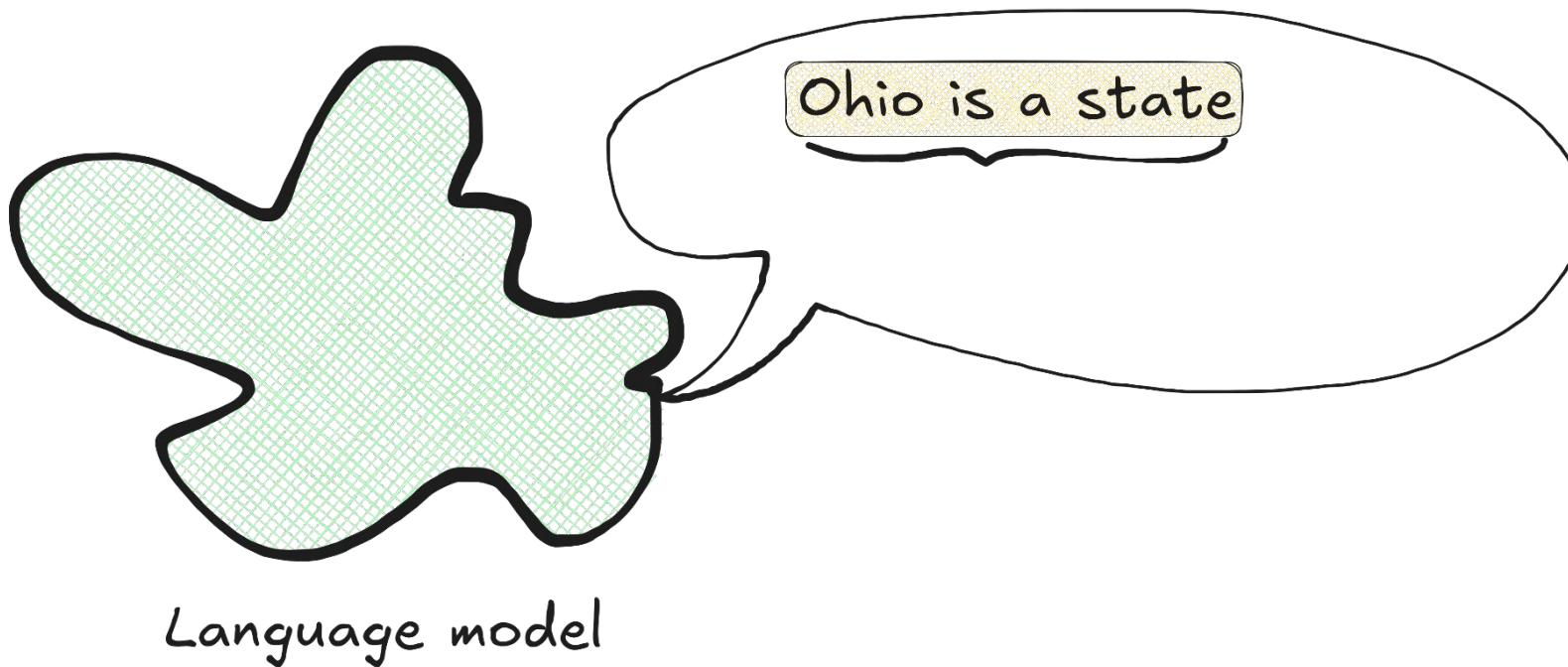
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



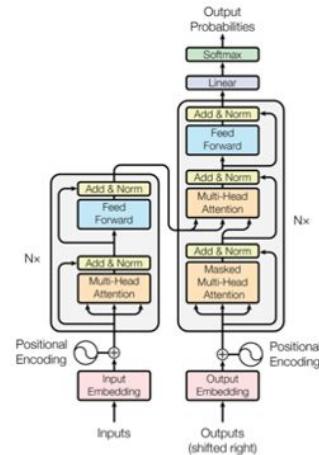
Building LMs: Neural Networks (1990s-2017)

Train complex neural networks to intelligently predict the next word.



Building LMs: Transformers (2017-Present)

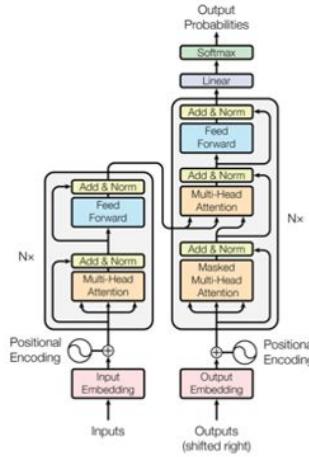
In 2017, Google introduced the **Transformer**



Building LMs: Transformers (2017-Present)

- Introduction of **two-stage pre-trained models** (BERT and GPT) transformed NLP.
- Reduced reliance on intermediate tasks (e.g., word segmentation, syntactic analysis).
- Enabled direct handling of final tasks like translation and text classification.
- Unified technical approaches across various NLP subfields, setting the stage for future innovations.
- More on this later...

The LLM Showdown (Age of Transformers)



The LLM Showdown (Age of Transformers)

Transformer

Attention Is All You Need

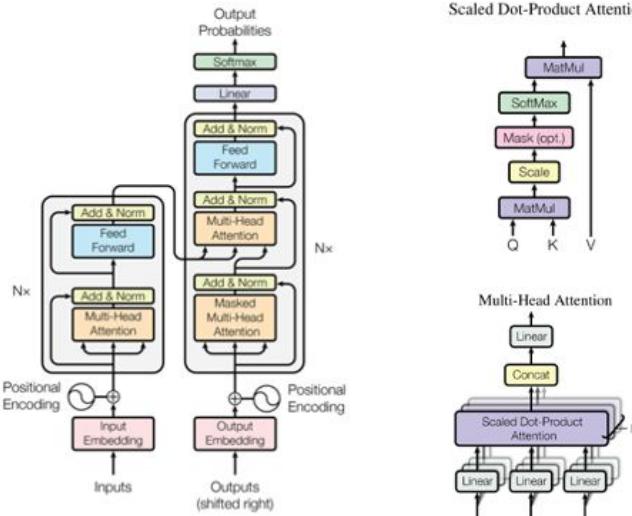


Figure 1: The Transformer - model architecture.



Transformer
2017-06

The screenshot shows the title "Attention Is All You Need" at the top. Below it is the abstract, which reads:

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. In this work, we show that an architecture based on multi-head attention layers can outperform these models with a much simpler and more efficient architecture. Specifically, our model achieves state-of-the-art results on the Penn Treebank and PTB2 language modeling datasets, while being more robust to changes in the architecture and hyperparameters. We also show that our model can be trained end-to-end on a single GPU in a few days, making it feasible to train very large models. Our model is also able to generate high-quality text, as evidenced by its performance on the IWSLT14MT task. Finally, we show that our model can be used as a language representation model, achieving state-of-the-art results on the IWSLT14MT task. Our model is also able to generate high-quality text, as evidenced by its performance on the IWSLT14MT task. Finally, we show that our model can be used as a language representation model, achieving state-of-the-art results on the IWSLT14MT task.

Using Transformer as Language Representation Models

detail. Niki designed, implemented, tested and evaluated countless model variants in our original codebase and tensor2tensor. Linzen also experimented with novel model variants, was responsible for our initial codebase and efficient inference and visualization. Lukasz and Adina spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.
*Work performed while at Google Brain.
**Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

The LLM Showdown (Age of Transformers)

GPT



Improving Language Understanding by
Generative Pre-Training

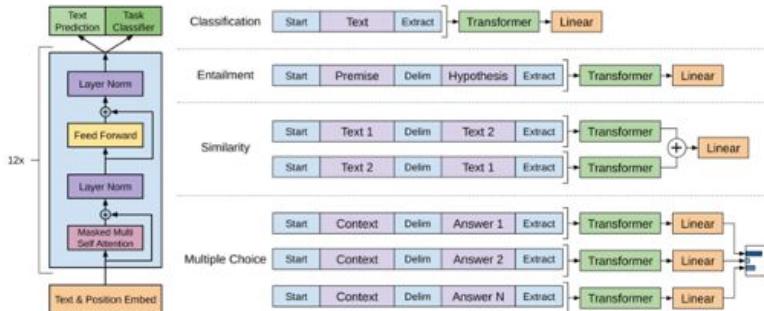


Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

GPT
2018-06

Improving Language Understanding
by Generative Pre-Training

Alec Radford
OpenAI
alecopenai.com Karthik Narasimhan
OpenAI
karthiknopenai.com Tim Salimans
OpenAI
tindopenai.com Ilya Sutskever
OpenAI
ilyasopenai.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

- **Decoder-only transformer architecture for autoregressive text generation**
- **Processes text sequentially with a left-to-right context.**
- **Generative pre-training over large corpus**

Proprietary. Work in progress.

The LLM Showdown (Age of Transformers)

BERT

BERT: Pre-training of Deep Bidirectional Transformer for Language Understanding

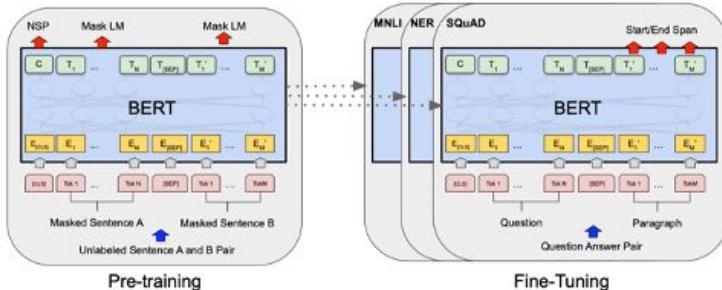


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).



BERT
2018-10

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin, mingweichang, kentonl, kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for Bidirectional Encoder Representations from Transformers. Unlike most language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations that can later be finetuned on both

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *finetuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The finetuning approach, such as the Generative Pre-trained Transformer (OpenAI

- Encoder-only transformer architecture.
- Optimized for understanding and interpreting text with bidirectional context.

2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationship between two sentences. They also include more holistic tasks, such as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Yi-jung Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

porate context from both directions. In this paper, we improve the fine-tuning based approaches by proposing BERT: Bidirectional Encoder Representations from Transformers. BERT alleviates the previously mentioned challenges by using a simple yet effective masked language model (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

The LLM Showdown (Age of Transformers)

GPT2



GPT2
2019-02

Language Models are Unsupervised Multitask Learners

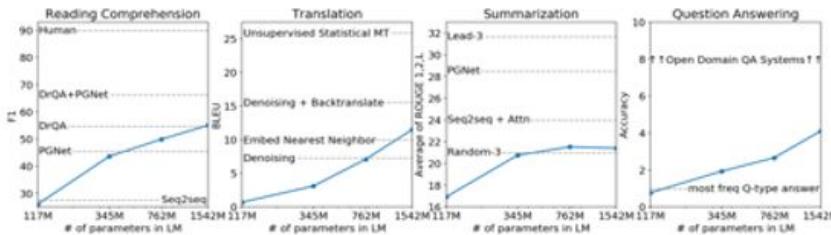


Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Language Models are Unsupervised Multitask Learners

Alec Radford ^{1,2} Jeffrey Wu ^{1,2} Rewon Child ¹ David Luan ¹ Dario Amodei ^{1,2} Ilya Sutskever ^{1,2}

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models can learn to perform these tasks via explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the untrained generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate those behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But as the scope of AI applications grows, so does the demand for systems that can handle multiple tasks at once.

Parameters

GPT-1 (117 M) → GPT-2 (1.5 B)

- Minimal architectural changes
- Context length

GPT-1 (512 tokens) → GPT-2 (1024 tokens)

et al., 2018). Yet these systems are brittle and sensitive to slight changes in the data distribution (Reich et al., 2018) and task specification (Karpukhin et al., 2017). Current systems are better characterized as narrow experts rather than

¹ Equal contribution. ²OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford (alec@openai.com).

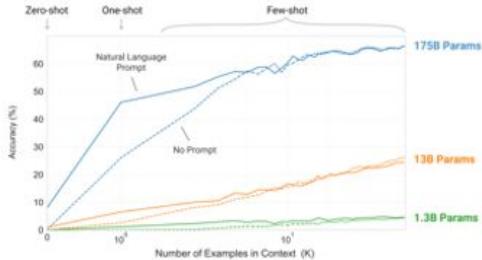
multitask training may be used just as many effective training pairs to realize its promise with current approaches. It will be very difficult to continue to scale the creation of datasets and the design of objectives to the degree that may be required to truly force our way there with current techniques. This work is also showing additional benefits for performing multitask learning.

The current best performing systems on language tasks

The LLM Showdown (Age of Transformers)

GPT3

Language Models are Few-Shot Learners



175 B!

Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{heads}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-6}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-6}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-6}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-6}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-6}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-6}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-6}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-6}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

GPT3
2020-05

arXiv:2005.14165v4 [cs.CL] 22 Jul 2020

Language Models are Few-Shot Learners

Tom B. Brown*, Benjamin Mann*, Nick Ryder*, Melanie Subbiah*, Jared Kaplan*, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramach, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Heer, Mark Chen, Eric Sigler, Maltez Litwin, Scott Gray, Benjamin Chen, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei

OpenAI

Abstract

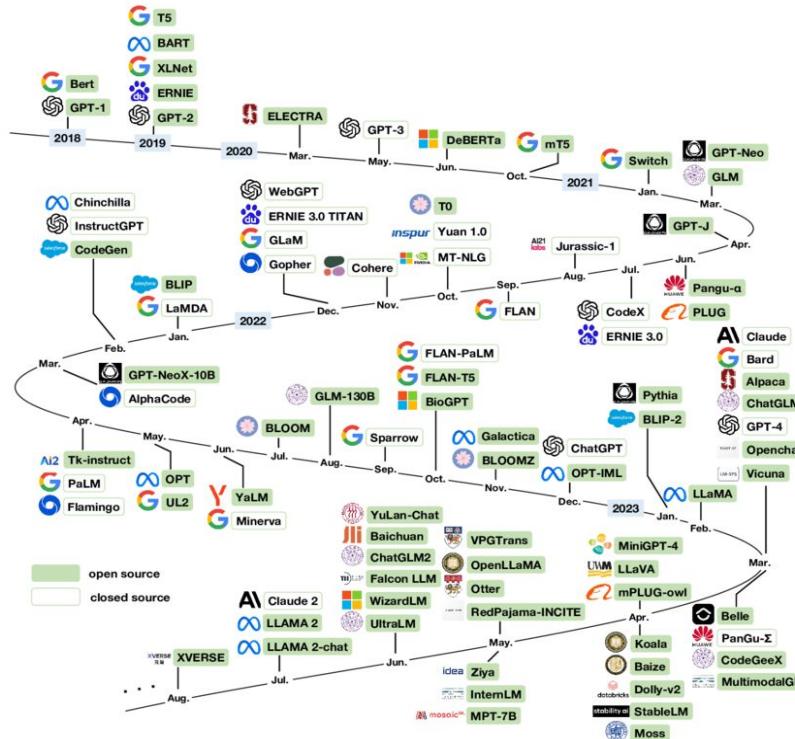
Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-specific, such models often still require thousands of examples to perform well, demanding thousands of hours of computation. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle with. In this paper, we show that large language models can also learn to perform complex, few-shot performance, sometimes even matching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, on a large, diverse dataset of text, and then evaluate its performance on a variety of tasks in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves state-of-the-art performance on a variety of benchmarks, including some very difficult and cluttered tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, GPT-3 exhibits significant performance degradation on a variety of other benchmarks, including some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution
†Johns Hopkins University, OpenAI
Author contributions listed at end of paper.

The LLM Showdown (Age of Transformers)

"How many Transformer models are out there?"

Trying to keep up with the new LLM
models dropping every few days



"Yes"

The ChatGPT Effect!



November 2022



ChatGPT



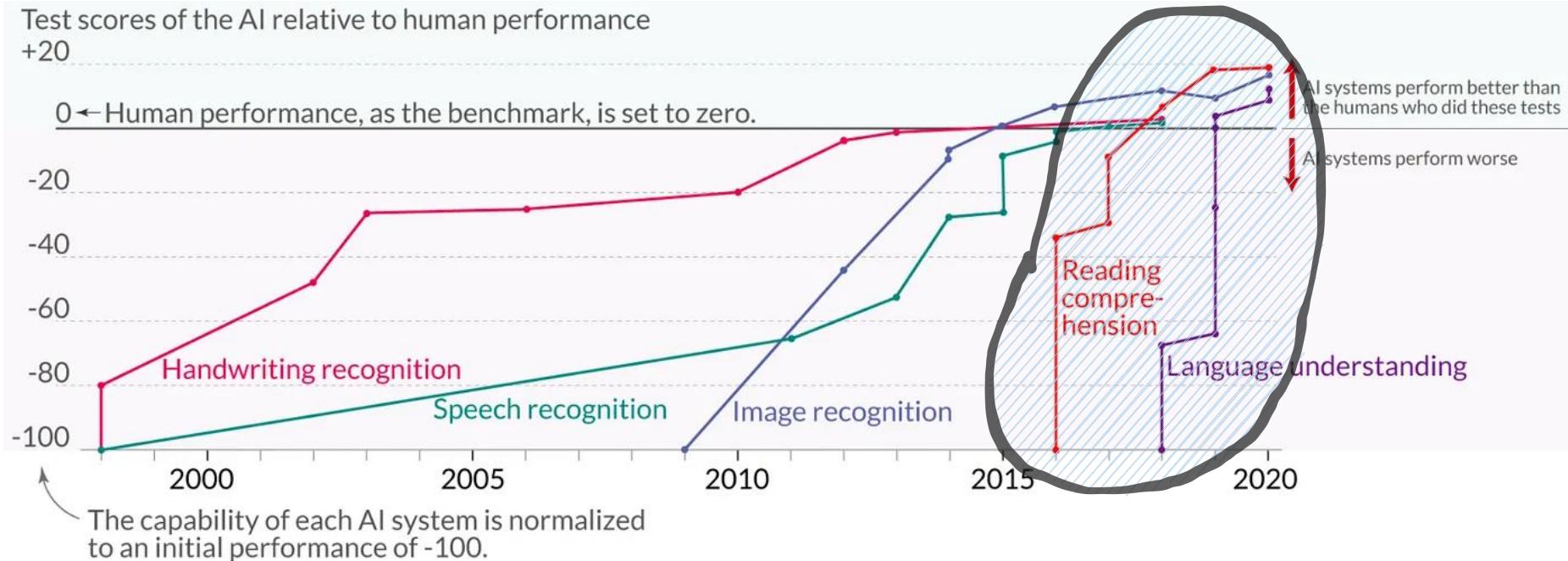
Tülu3



The ChatGPT Effect!

Test scores of the AI relative to human performance
+20

0 ← Human performance, as the benchmark, is set to zero.



Comprehensive LLM Journey:

- Explore foundational concepts in NLP to cutting-edge topics in LLMs.

Industry Alignment:

- Learn methods and tools that are widely used in industry (e.g., Transformers, fine-tuning LLMs, RL optimizations).

Skill Enhancement:

- Gain hands-on experience with advanced models and optimization techniques in Generative AI.

Utility for Students:

- Understand the historical evolution of LLMs, providing context for current trends.
- Grasp core NLP concepts (statistical modeling, smoothing, word embeddings) to build a strong theoretical base.

Real-World Application:

- Apply foundational concepts to solve practical language processing problems, such as text classification and sentiment analysis.

Core Learning Outcomes:

- Learn key neural network architectures (CNNs, RNNs, LSTMs) essential for modern NLP tasks.
- Learn sequence-to-sequence models and backpropagation, critical for building custom language models.

Utility in Practice:

- Develop a strong intuition for designing and training neural models, which is applicable in various AI projects.
- Understand advanced training techniques (attention, optimizers, normalization) to improve model performance in real-world applications.

About this Course

Transformers & LLM Training (Modules 4 & 5)

Concept Mastery:

- Learn the transformer architecture
- Explore tokenization methods and their impact on model performance.

Practical Benefits:

- Gain insights into LLM training pipelines: pre-training, fine-tuning, and in-context learning.
- Understand preference optimization techniques (e.g., RLHF) to refine model responses—vital for building user-centric AI applications.

About this Course

LLM Optimization & Reasoning (Modules 6 & 7)

Optimization Techniques:

- Learn strategies for inference optimization (batching, quantization, caching) that reduce cost and latency in production systems.
- Explore PEFT fine-tuning and distillation to make models more practical and deployable.

Enhanced Reasoning Skills:

- Develop abilities to use search techniques, test-time scaling, and chain-of-thought reasoning, enhancing model interpretability.

About this Course

Advanced Architectures & Emerging Applications
(Modules 8, 9 & 11)

Future-Focused Learning:

- Dive into advanced architectures such as multimodal LLMs, Mixture of Experts, and diffusion models.
- Understand the integration of retrieval mechanisms in RAG systems to build smarter and more context-aware applications.

Cross-Disciplinary Utility:

- Gain exposure to diffusion models and vision-language tasks, expanding your expertise beyond pure NLP.

About this Course

Advanced Architectures & Emerging Applications
(Modules 8, 9 & 11)

Innovation & Research:

- Prepare for research or entrepreneurial ventures by understanding emerging trends and technologies that are reshaping the AI landscape.