

CS 435-L1

Lecture 2

Adnan Masood, PhD.

Administrivia

The Evolution of LLM

Recap Quiz & Coding Exercises

The Ethical Challenges with LLM

Transformer Architecture

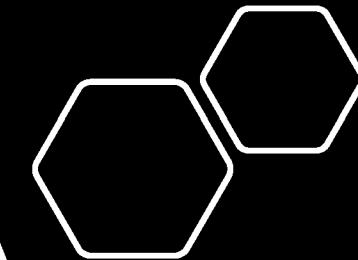
Coding Assignment - Tokenizer

Week-Wise Schedule (Tentative)

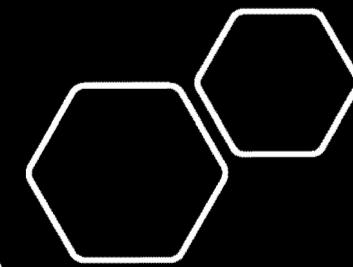
Spring 2024 Weekly Schedule*



Week	Dates	Topics	Assessments
1	Jan 8 -- 12, 2024	Introduction to Generative AI Large Language Models (LLMs)	Assignment 1
2	Jan 15 -- 19, 2024	LLM Architecture Libraries for LLMs	Recap Quiz 1 Assignment 2
3	Jan 22 -- 26, 2024	Guest Lecture on Generative AI Foundation Models	Assignment 3
4	Jan 29 -- Feb 2, 2024	Ethics of AI AI Security Fundamentals	Recap Quiz 2 Assignment 4
5	Feb 5 -- 9, 2024 Kashmir Day: Feb 5	AI Governance Frameworks AI Regulations	Recap Quiz 3 Assignment 5
6	Feb 12 -- 16, 2024	Vulnerabilities in LLMs Security Issues in LLMs	Recap Quiz 4 Assignment 6
7	Feb 19 -- 23, 2024	Advanced Security Testing for LLMs LLM Security Frameworks	Recap Quiz 5 Assignment 7
8	Feb 26 -- Mar 1, 2024	Ethical Considerations in LLM Development Ethical Considerations in LLM Deployment	Recap Quiz 6 Assignment 8
9	Mar 4 -- 8, 2024	Guest Lecture on AI Governance Cloud Computing for AI	Reflection Discussion Assignment 9
10	Mar 11 -- 15, 2024 1st Ramadan: Mar 11	LLM Security in AWS LLM Security in Google Cloud Platform (GCP)	Recap Quiz 7 Assignment 10
11	Mar 18 -- 22, 2024 Conference Days: Mar 22--24 (No Classes)	LLM Security in Microsoft Azure Open-Source LLM Security	Recap Quiz 8 Assignment 11
12	Mar 25 -- 29, 2024 Last Day to Withdraw: Mar 29	Hugging Face and LLM Security Enterprise Aspects of LLM Security	Recap Quiz 9 Assignment 12



13	Apr 1 -- 5, 2024	Future Trends in LLM Security Emerging Challenges in AI Security	Recap Quiz 10
14	Apr 8 -- 12, 2024 Eid ul Fitr: Apr 10 -- 12	Final Project Presentations	
15	Apr 15 -- 19, 2024	Final Project Presentations	
16	Apr 22 -- 26, 2024 Last Day of Classes: Apr 26	Course Recap and Future Directions Career Paths in AI Security and Ethics	
Reading Days	Apr 27 -- 29, 2024		
Exam Period	April 30, May 2--3, and May 6--8, 2024 Labor Day: May 1, 2024 Last Date to File Petition for Incomplete Grade: May 8, 2024		
End of Term	May 9 -- 11, 2024		



MEET THE RESEARCH ASSISTANTS

KARACHI

HABIB
UNIVERSITY
KARACHI



Shafaq Fatima Mughal is a Research Assistant in Computer Science at Habib University and a Fulbright Scholar. With a strong academic background in Computer Science and a minor in Mathematics, she specializes in Data Science and Artificial Intelligence. Her work involves mentoring students on AI research, drafting grant proposals, and developing innovative assignments. Shafaq's research contributions include sentiment analysis for Urdu using transformer models, and her accolades include recognition for her final year project and academic excellence. She is passionate about advancing technology and education through impactful research and mentorship.



Omema Rizvi is a Research Assistant in Computer Science at Habib University, with a strong academic foundation in Computer Science and a demonstrated commitment to leadership in STEM. As a treasurer for both the Women in Science and Engineering (WiSE) and Habib University Artificial Intelligence Club (HUAIC), she has fostered innovation and collaboration within her academic community. Omema's experience spans research, student mentorship, and community engagement, making her a dedicated advocate for advancing technology and empowering underrepresented groups in science and engineering.

Office Hours

Every Tuesday – 6 AM US EST
/ 4 PM Pakistan Time via
Teams.



On-Demand (send me an
email with specific
query/agenda and cc' the
RAs)





4 Stages of Psychological Safety

01 Inclusion Safety

02 Learner Safety

03 Contributor Safety

04 Challenger Safety



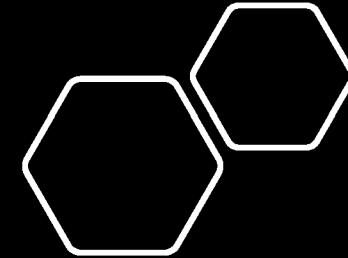
[Overview](#)[NotebookLM Plus](#)

Think Smarter, Not Harder

The ultimate tool for understanding the information that matters
most to you, built with Gemini 2.0

[Try NotebookLM](#)

Your Personalized AI Research Assistant



Administrivia

The Evolution of LLM

Recap Quiz & Coding Exercises

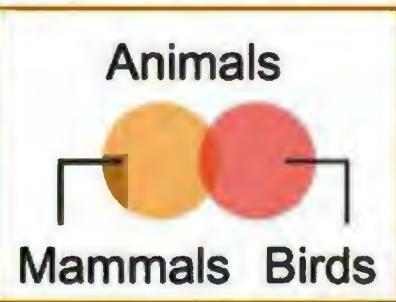
The Ethical Challenges with LLM

Transformer Architecture

Coding Assignment - Tokenizer

What are the five tribes?

Symbolists



Use symbols, rules, and logic to represent knowledge and draw logical inference

Favored algorithm

Rules and decision trees

Bayesians

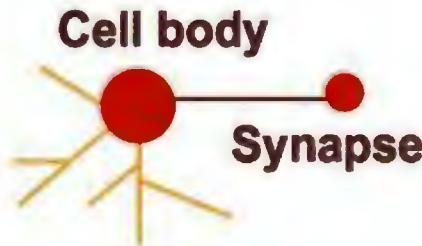


Assess the likelihood of occurrence for probabilistic inference

Favored algorithm

Naive Bayes or Markov

Connectionists

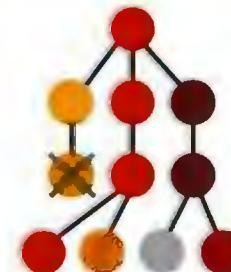


Recognize and generalize patterns dynamically with matrices of probabilistic, weighted neurons

Favored algorithm

Neural networks

Evolutionaries

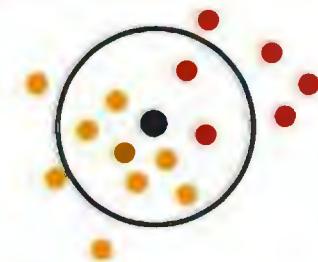


Generate variations and then assess the fitness of each for a given purpose

Favored algorithm

Genetic programs

Analogizers



Optimize a function in light of constraints ("going as high as you can while staying on the road")

Favored algorithm

Support vectors

Defining Generative AI

To understand generative artificial intelligence (GenAI), we first need to understand how the technology builds from each of the AI subcategories listed below.

Artificial Intelligence

The theory and methods to build machines that think and act like humans.



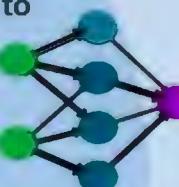
Machine Learning

The ability for computers to learn from experience or data without human programming.



Deep Learning

Mimics the human brain using artificial neural networks such as **transformers** to allow computers to perform complex tasks.



Generative AI

Generates new text, audio, images, video or code based on content it has been **pre-trained on**.



ChatGPT Midjourney Bard

Different Views of Generative AI

Basic

Intermediate

Advance

Frontier



Gemini



Hugging Face

LLM fine-tuning at a high level

LLM fine-tuning

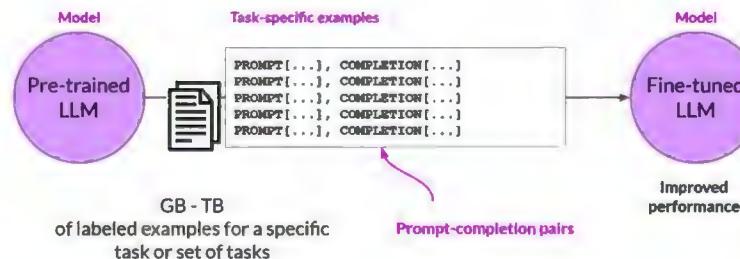
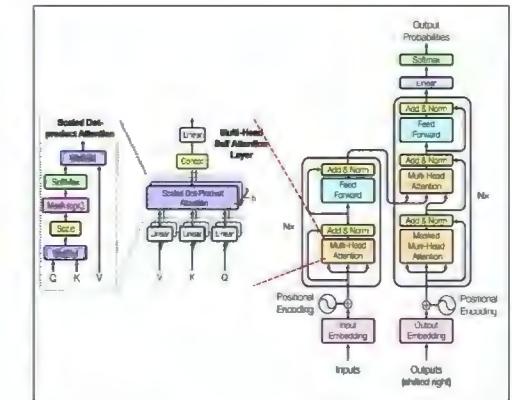
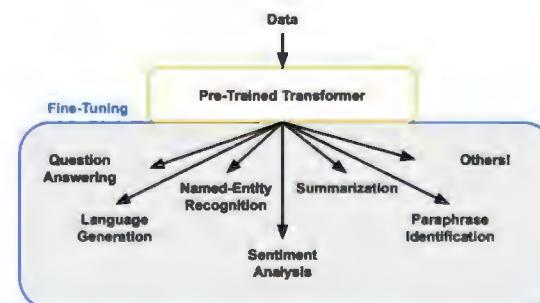
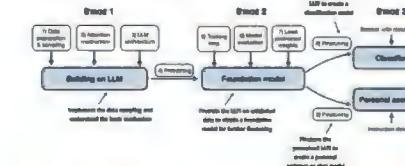


Figure 3. Capabilities of an LLM after the fine tuning



Build a Large Language Model from Scratch



A truly great business must
have an enduring “moat” that
protects excellent Returns on
Invested Capital.

Warren Buffett
2007 Shareholder Letter





Present

**More Difficult +
More Time Required +
Deeper Moat**

Practical

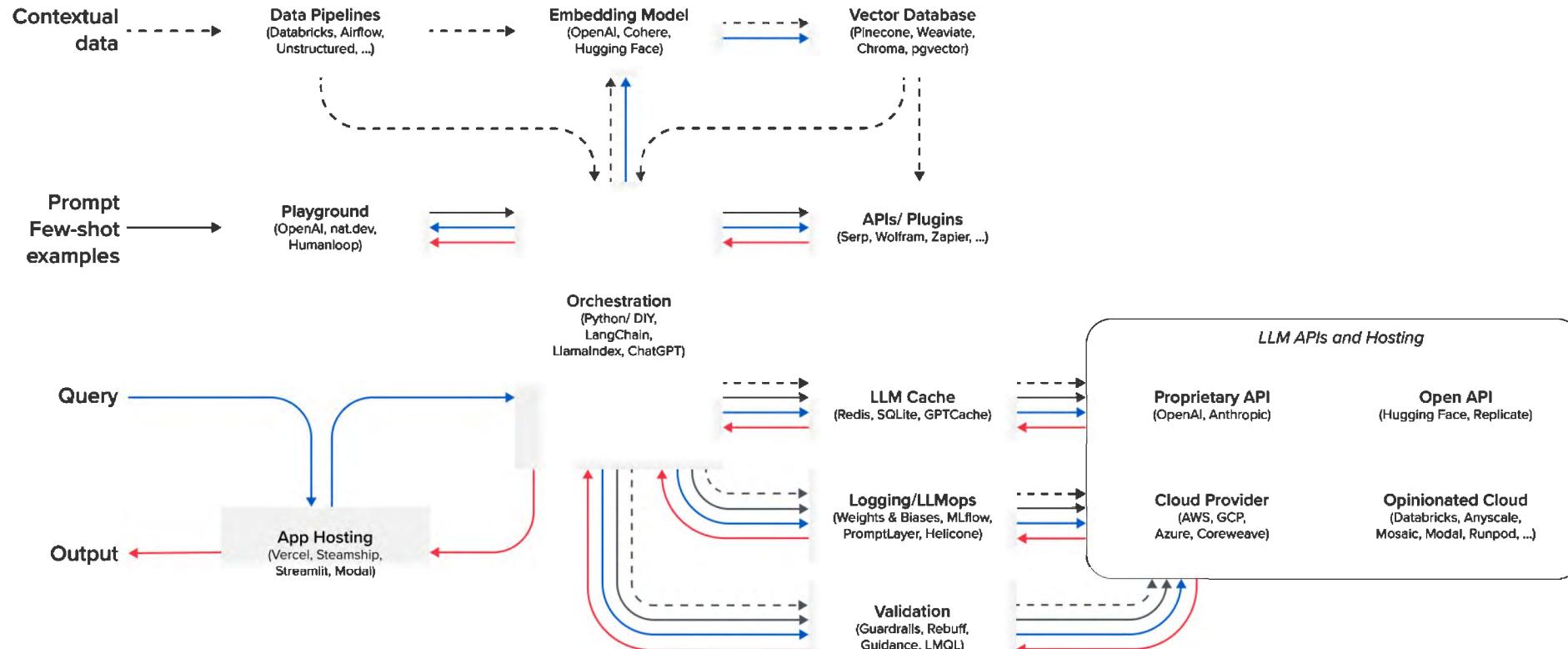
Powerful

Starting with AI

AI Maturity

Ideal Time to Invest

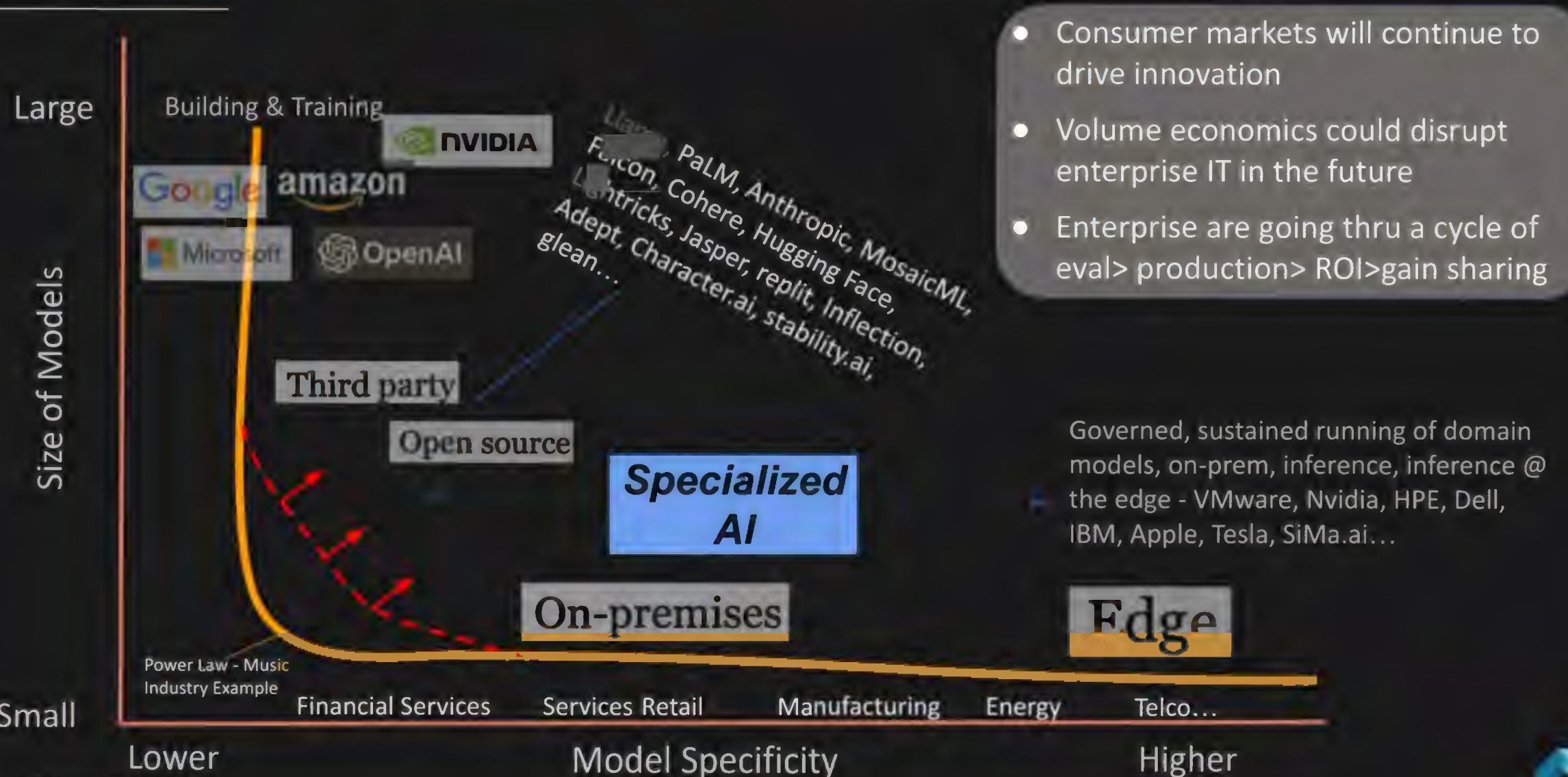
Emerging LLM App Stack



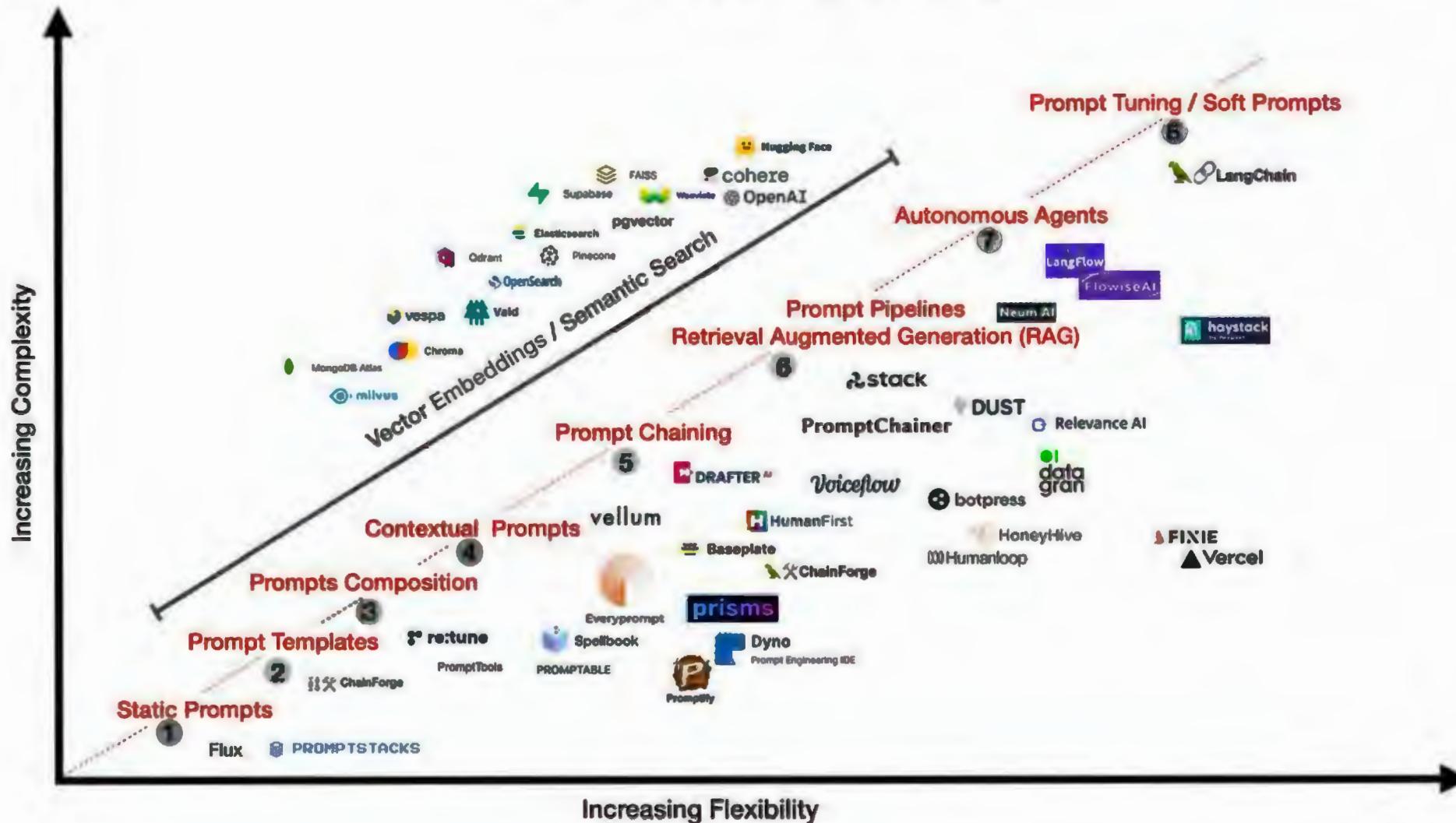
LEGEND

- Gray boxes show key components of the stack, with leading tools/systems listed
- Arrows show the flow of data through the stack
- - - → Contextual data provided by app developers to condition LLM outputs
- Prompts and few-shot examples that are sent to the LLM
- Queries submitted by users
- Output returned to users

Power Law Distribution of Gen AI



Emerging LLM Application Architecture





prerequisites

adjective

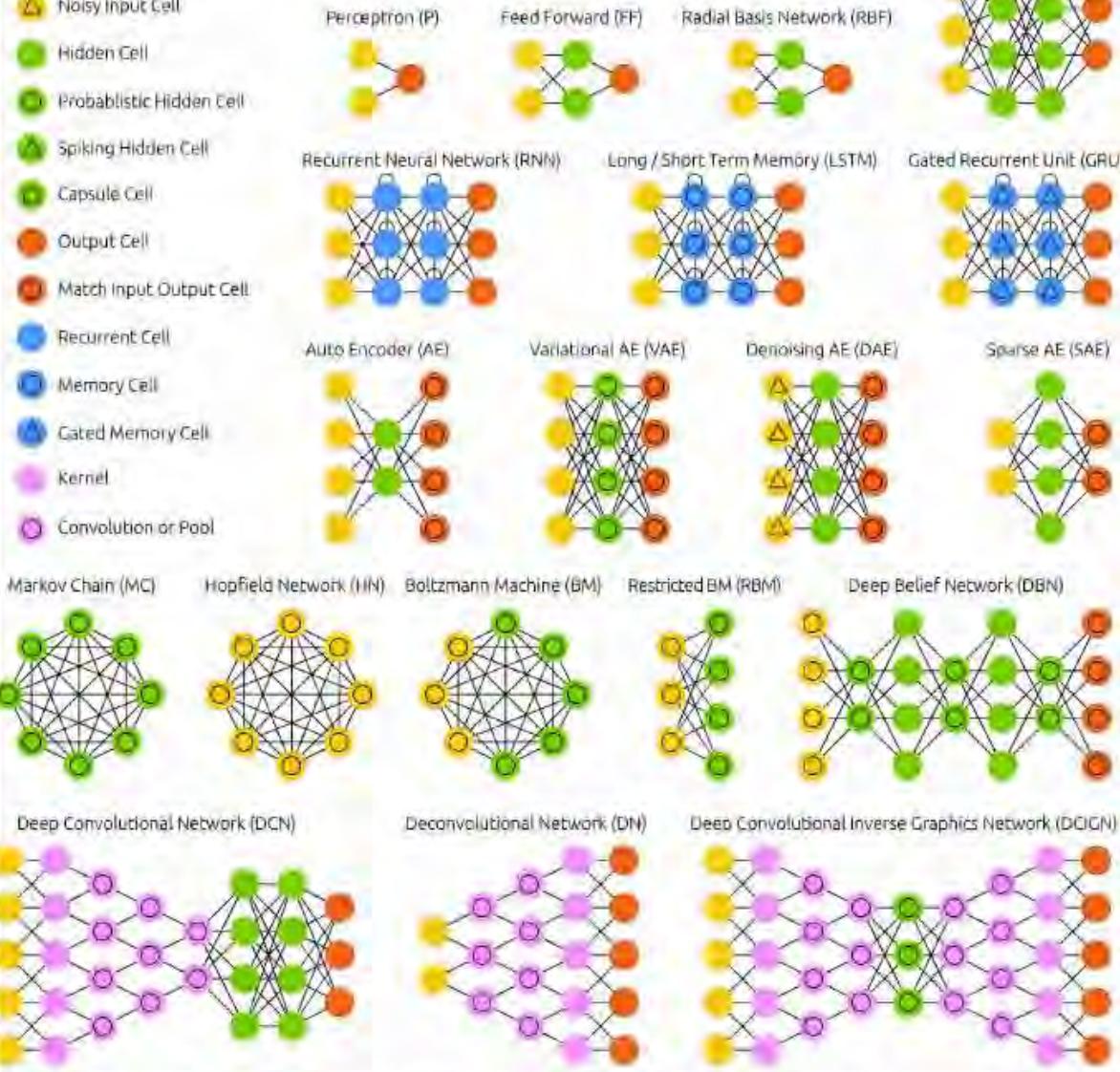
Definition : Required as a prior condition.

Example : the student must have the prerequisite skills

A mostly complete chart of
Neural Networks

©2019 Fjodor van Veen & Stefan Leijen asimovinstitute.org

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input/Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool



<https://www.asimovinstitute.org/neural-network-zoo/>

All Courses

Grow your AI career with **foundational specializations** and skill-specific short courses taught by leaders in the field.



[Explore Courses](#)[AI Newsletter](#)[Community](#)[Resources](#)[Company](#)[Start Learning](#)[All Courses](#) > [Short Courses](#) > [AI Python for Beginners](#)[Short Course](#) [Beginner](#) [4 Hours 15 Minutes](#)

AI Python for Beginners

Instructor: Andrew Ng

[Enroll for Free](#)

What you'll learn

- ✓ Learn Python programming fundamentals and how to integrate AI tools for data manipulation, analysis, and visualization.
- ✓ Discover how Python can be applied in various domains such as business, marketing, and journalism to solve real-world problems and enhance efficiency through practical applications.
- ✓ Leverage AI assistants to debug code, explain concepts, and enhance your learning, mirroring real-world software development practices.

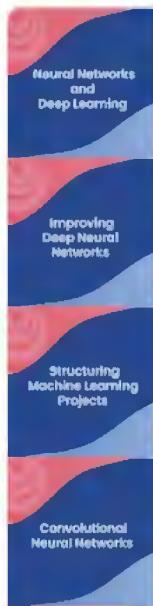
AI Python for Beginners

[Beginner](#)[4 Hours 15 Minutes](#)[35 Video Lessons](#)[25 Code Examples](#)[Instructor: Andrew Ng](#)[DeepLearning.AI](#)[Enroll for Free](#)

Python Programming Deep Learning Backpropagation Optimization Hyperparameter Tuning Machine Learning
Transfer Learning Multi-Task Learning Object Detection and Segmentation Facial Recognition System
Gated Recurrent Unit (GRU) Long Short Term Memory (LSTM) Attention Models Natural Language Processing



Syllabus



Course 1: Neural Networks and Deep Learning

In the first course of the Deep Learning Specialization, you will study the foundational concept of neural networks and deep learning.

Course 2: Improving Deep Neural Networks: Hyperparameter Tuning, Regularization, and Optimization

By the end, you will be familiar with the significant technological trends driving the rise of deep learning; build, train, and apply fully connected deep neural networks; implement efficient (vectorized) neural networks; identify key parameters in a neural network's architecture; and apply deep learning to your own applications.

Course 3: Structuring Machine Learning Projects

Week 1: Introduction to Deep Learning

Understand the significant technological trends driving deep learning development and where and how it's applied.

Week 2: Neural Networks Basics

Course 4: Convolutional Neural Networks

LSTM is dead. Long live Transformers!

Leo Dirac

12 Nov, 2019

Sea-ADL.org



<https://www.youtube.com/watch?v=S27pHKBEp30>



The busy person's
intro to LLMs

Andrej Karpathy

0:01 / 59:47 • Intro: Large Language Model (LLM) talk >

[1hr Talk] Intro to Large Language Models

▶ ▶ | 🔍 CC ⚙️ 🎞 🎞

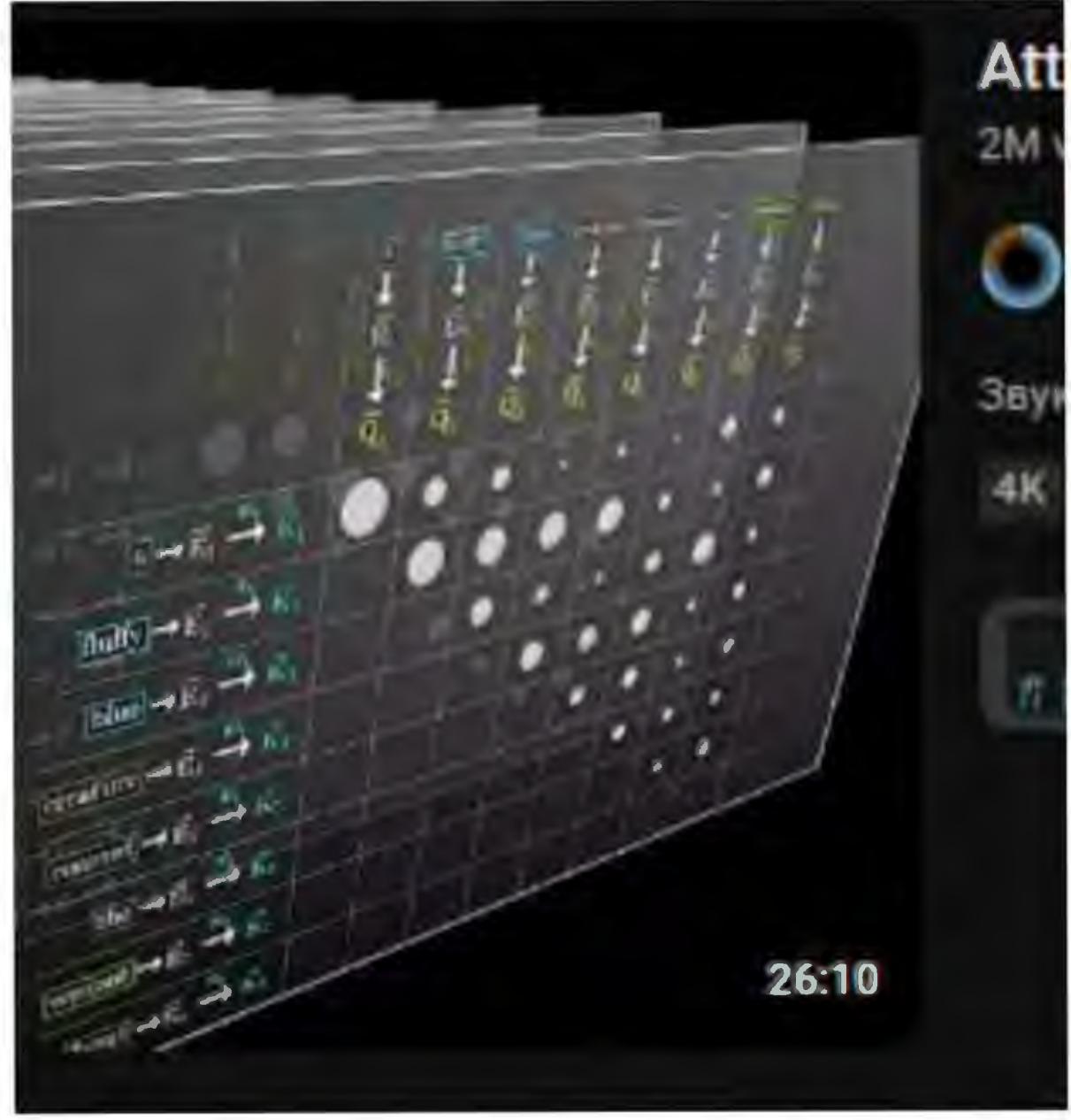
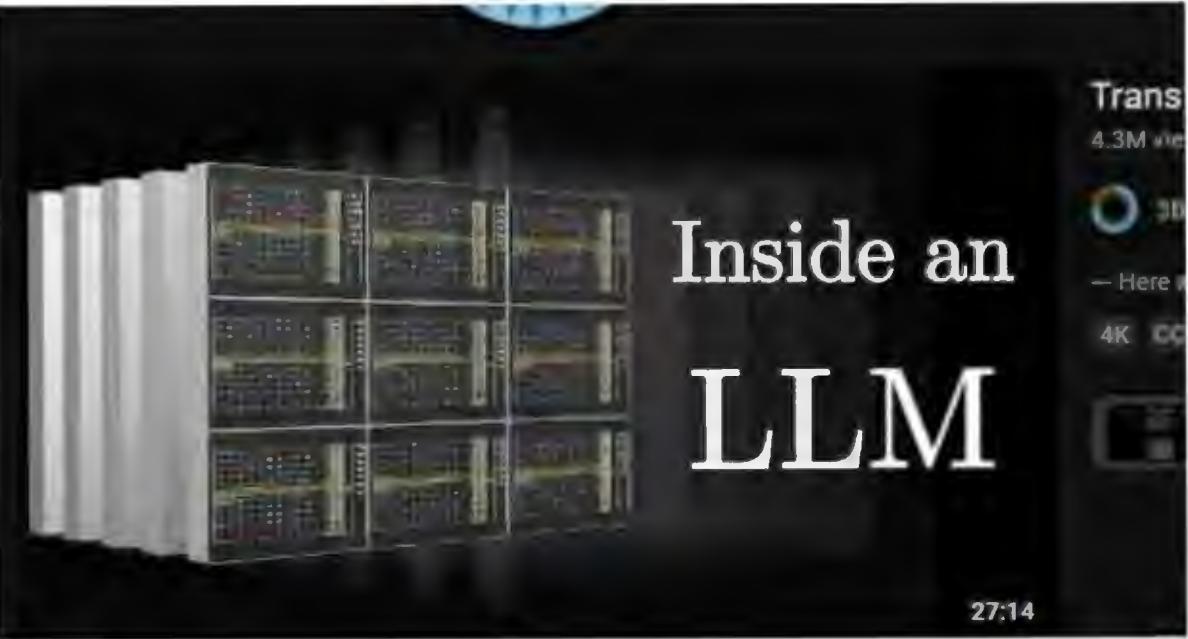
https://www.youtube.com/watch?v=zjkBMFhNj_g

A screenshot of a YouTube video player. The main image is a colorful, abstract illustration of Earth's atmosphere, densely packed with glowing icons representing various technologies and data points. To the right of the image is a white rectangular box containing the title "The busy person's intro to LLMs" and the speaker's name, "Andrej Karpathy", in italics. Below the main image is a video frame showing a man with short brown hair, identified as Andrej Karpathy. At the bottom of the screen are standard YouTube video controls: play/pause, volume, progress bar, and a list of video thumbnails. The overall theme is a technical introduction to Large Language Models (LLMs) presented in a visually appealing and accessible way.

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

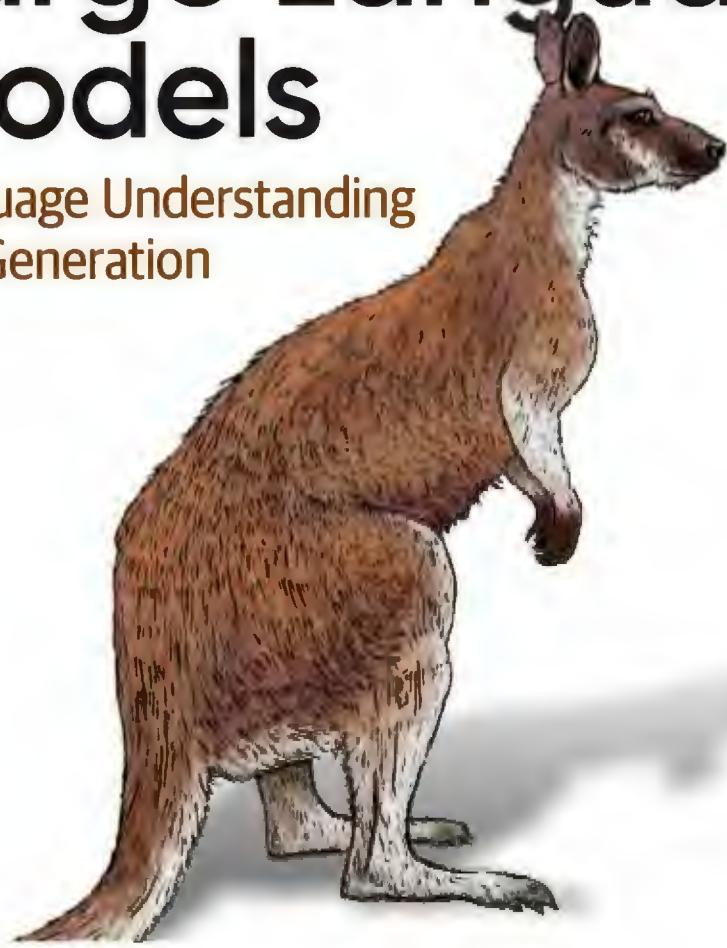


<https://www.youtube.com/watch?v=kCc8FmEb1nY>



Hands-On Large Language Models

Language Understanding
and Generation



Jay Alammar

Visualizing machine learning one concept at a time.
[@JayAlammar](#) on Twitter.

The Illustrated Transformer

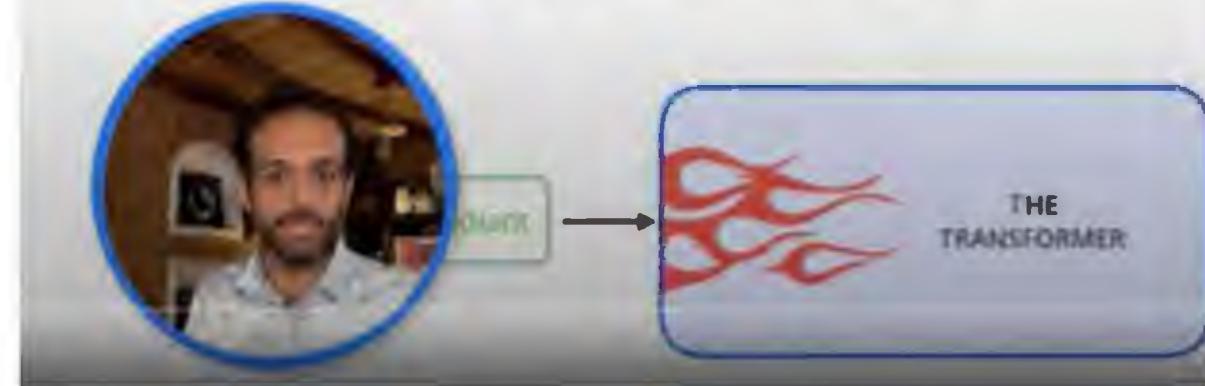
Discussions: [Hector Nunez](#) (60 points, 4 comments), [HAAHLM](#) (MachineLearning) (20 points, 0 comments)

Translations: [Chinese](#) (SimClef), [Armenian](#), [Hungarian](#), [Russian](#)

Watch [MIT's OpenCourseWare](#) [State of the Art](#) lecture referencing this post

A High-Level Look

Let's begin by looking at the model as a single black box. In a machine translation, input a sentence in one language, and output its translation in another.



LLMs Building Blocks



Administrivia

The Evolution of LLM

Recap Quiz & Coding Exercises

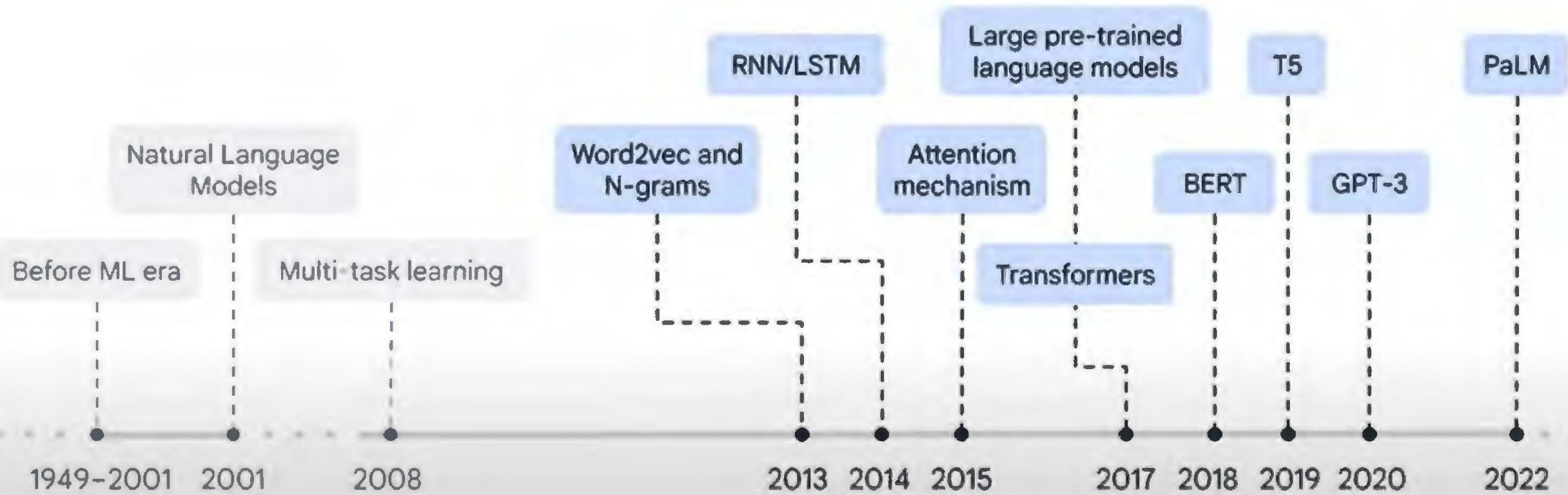
The Ethical Challenges with LLM

Transformer Architecture

Coding Assignment - Tokenizer

The Evolution of Large Language Models (LLMs) - From Perceptrons to GPT-4 and Beyond

Language modeling history



1. How many 2-grams (bigrams) are present in the following phrase:

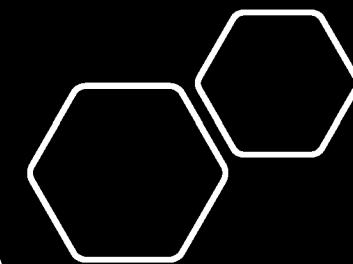
they visited New York last week

3

4

5

6





**Talk is cheap.
Show me the code.**

Linus Torvalds



cor·pus
/kôrpəs/

noun

1. a collection of written texts, especially the entire works of a particular author or a body of writing on a particular subject.

"the Darwinian corpus"

Similar: collection

compilation

body

entity

whole

aggregation

mass

2. ANATOMY

the main body or mass of a structure.

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

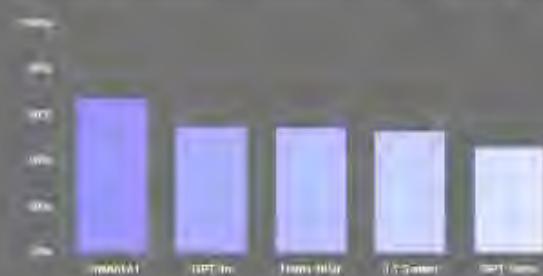
Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

LLM Leaderboard

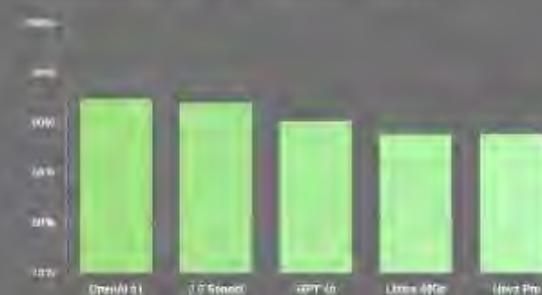
<https://www.vellum.ai/llm-leaderboard>

Top Models per Task

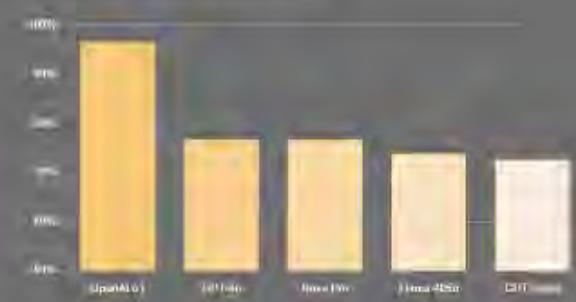
Best in Multitask Reasoning (MMLU) ^①



Best in Coding (Human Eval) ^②

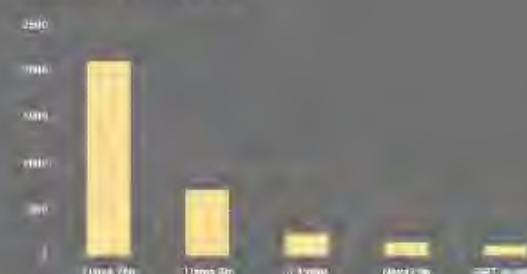


Best in Math (MATH) ^③



Fastest and Most Affordable Models

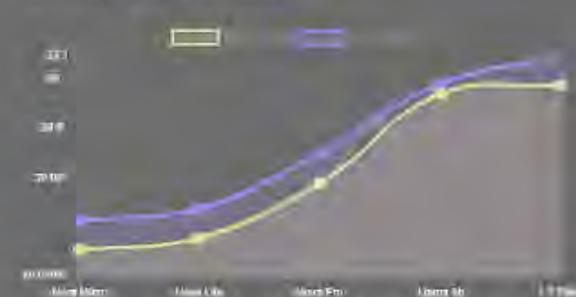
Fastest Models ^④



Lowest Latency (TTFT) ^⑤



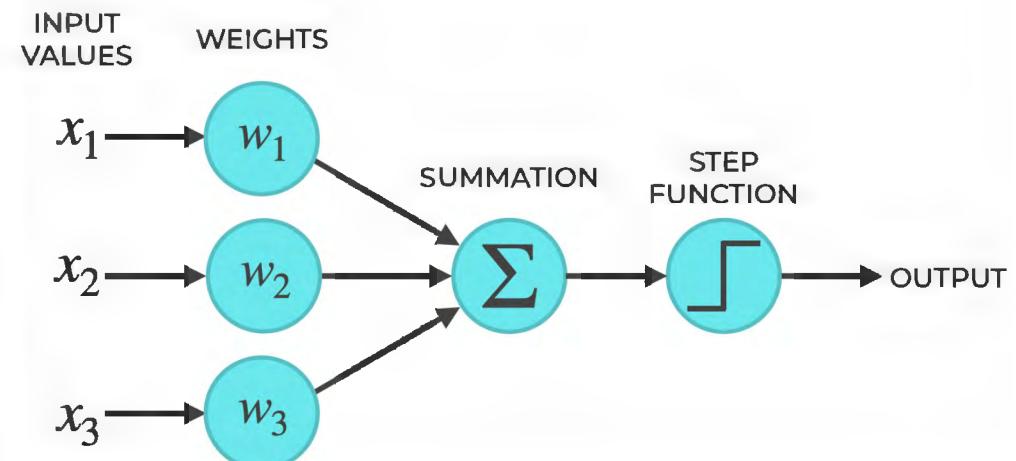
Cheapest Models ^⑥



Perceptrons: The Beginning (1958)

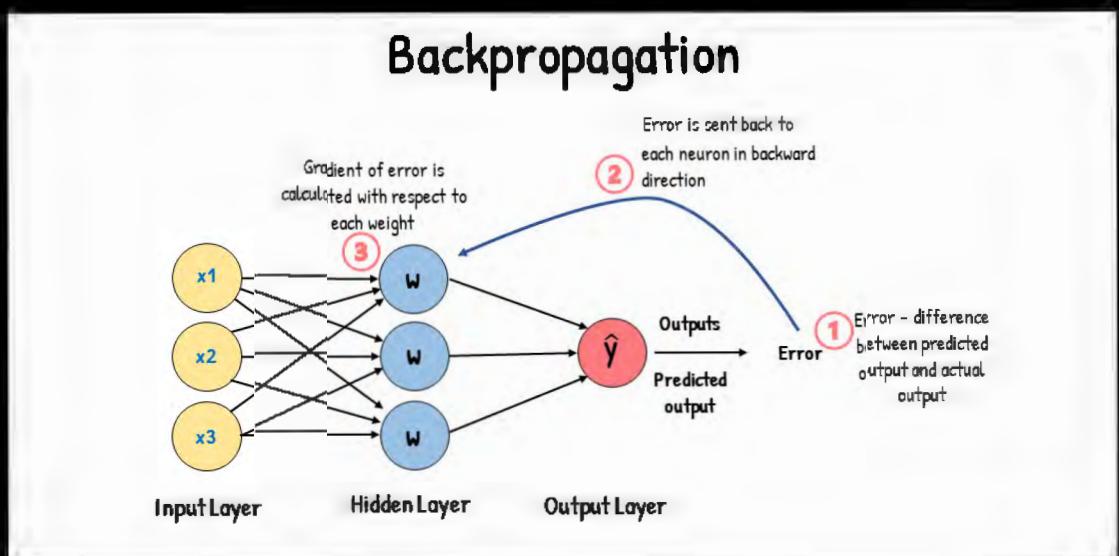
- Key Innovator: Frank Rosenblatt
- Paper: The Perceptron - A Probabilistic Model for Information Storage and Organization in the Brain
- Concept: Basic neural network for pattern recognition
- Limitation: Could not solve non-linear problems like XOR.

PERCEPTRONS, EXPLAINED



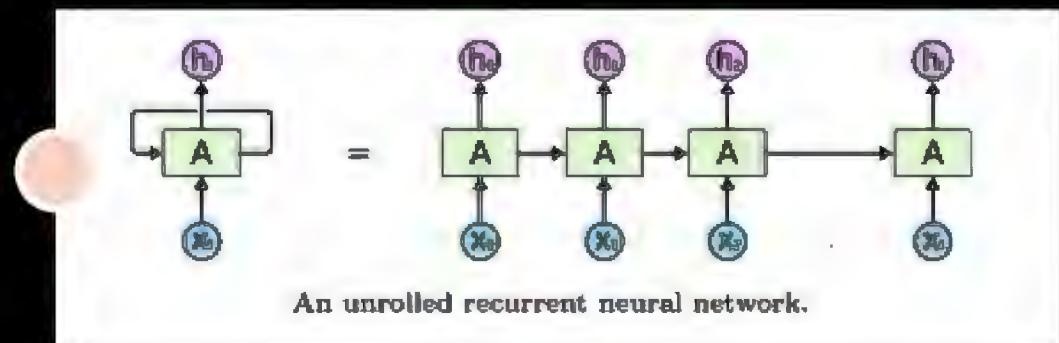
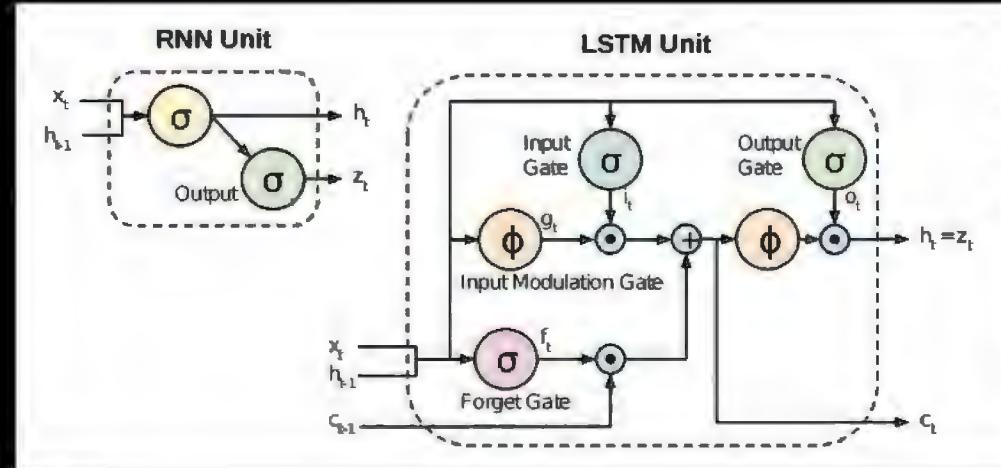
The AI Winter and Backpropagation (1980s)

- Challenges: Highlighted by Marvin Minsky and Seymour Papert in Perceptrons (1969)
- Rediscovery: Backpropagation by David Rumelhart, Geoffrey Hinton, and Ronald Williams
- Paper: Learning Representations by Back-Propagating Errors (1986)
- Impact: Revived interest in neural networks.

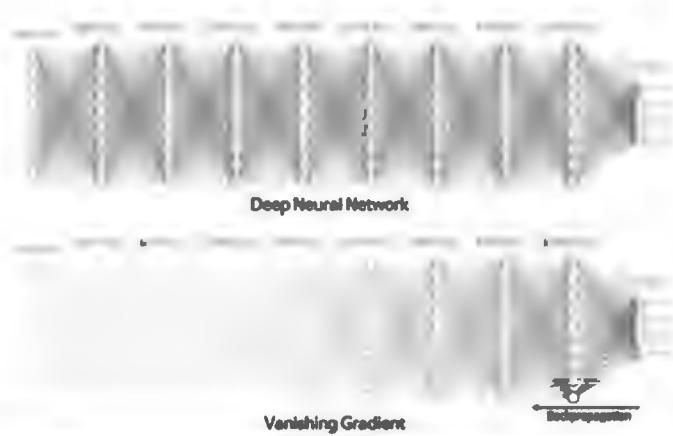
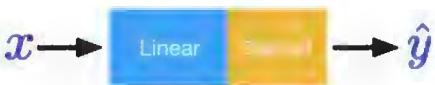


Recurrent Neural Networks (RNNs) and LSTMs (1990s)

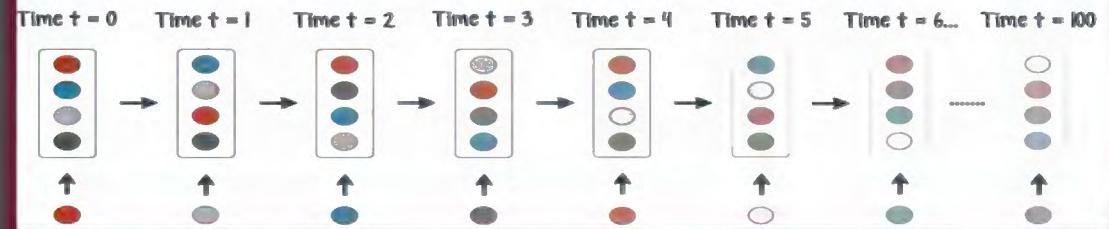
- Innovators: Sepp Hochreiter and Jürgen Schmidhuber
- Paper: Long Short-Term Memory (1997)
- Concept: RNNs introduced loops for sequential data; LSTMs addressed the vanishing gradient problem.
- Applications: Speech recognition, language modeling.



Sigmoid: Vanishing Gradient Problem

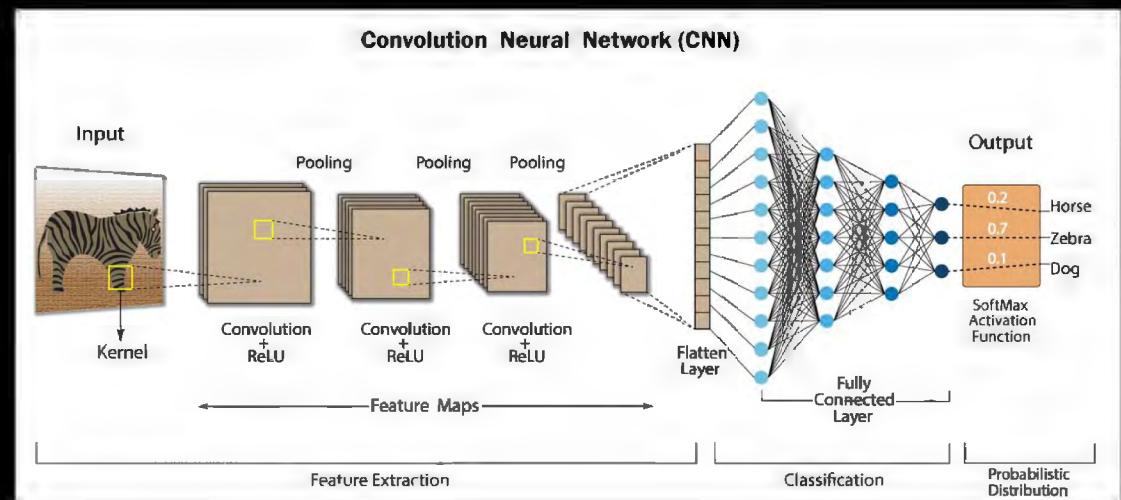


Decay of information through time



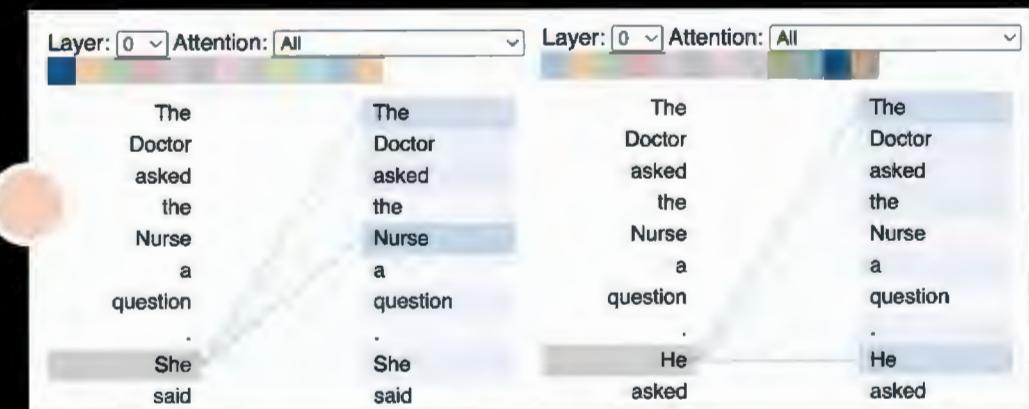
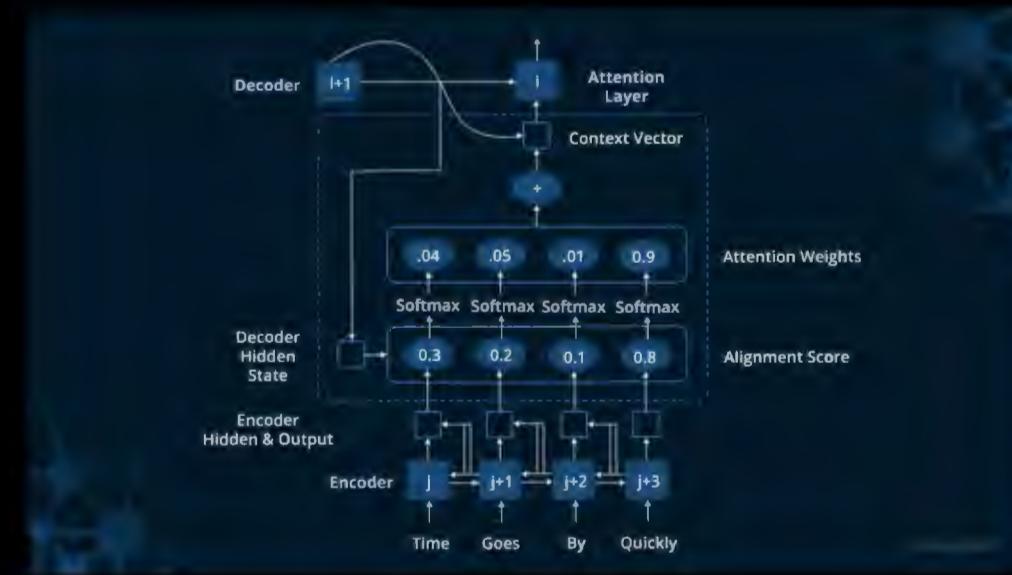
Convolutional Neural Networks (CNNs) (1980s–2010s)

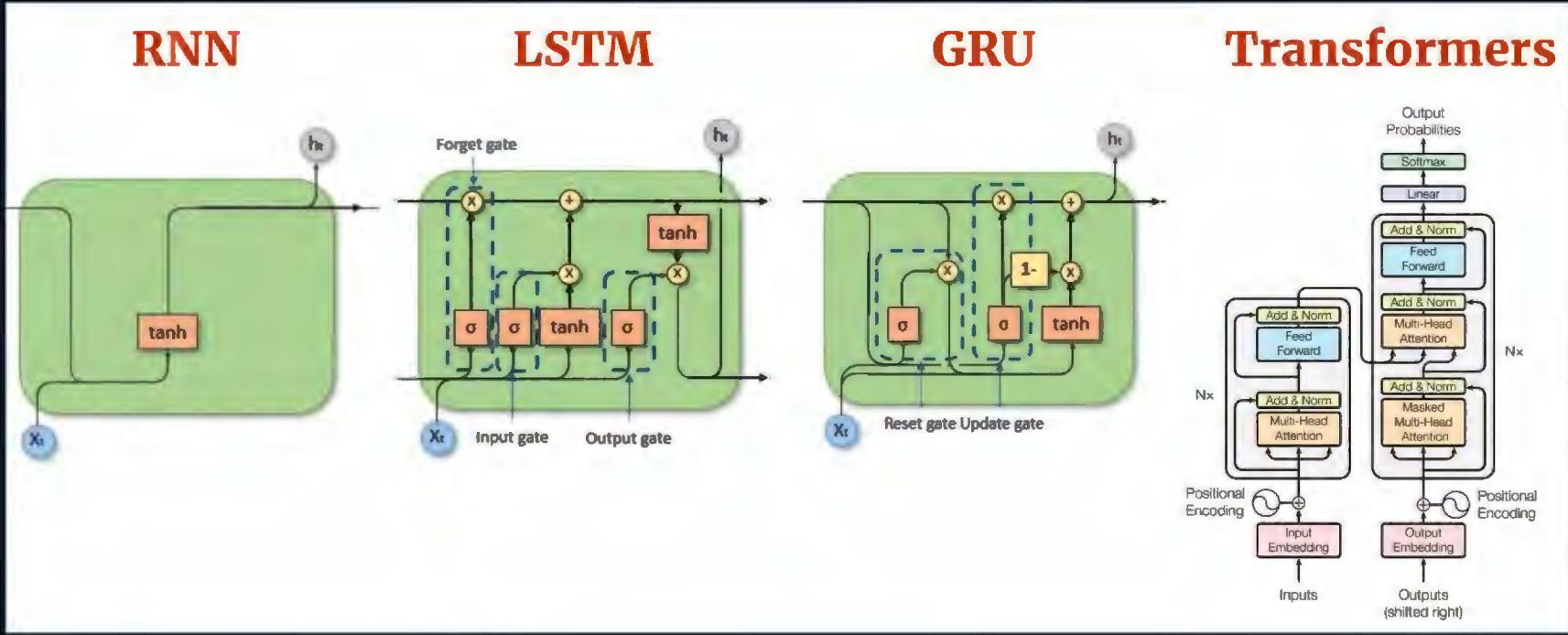
- Key Innovator: Yann LeCun
- Paper: Gradient-Based Learning Applied to Document Recognition (1998)
- Concept: CNNs excelled in image recognition through local receptive fields and weight sharing.
- Influence: Inspired architectural advancements for NLP.



Attention Mechanisms: A Turning Point (2014)

- Key Researchers: Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio
- Paper: Neural Machine Translation by Jointly Learning to Align and Translate
- Concept: Attention mechanisms allowed models to focus on relevant input parts.
- Impact: Breakthroughs in translation and summarization.

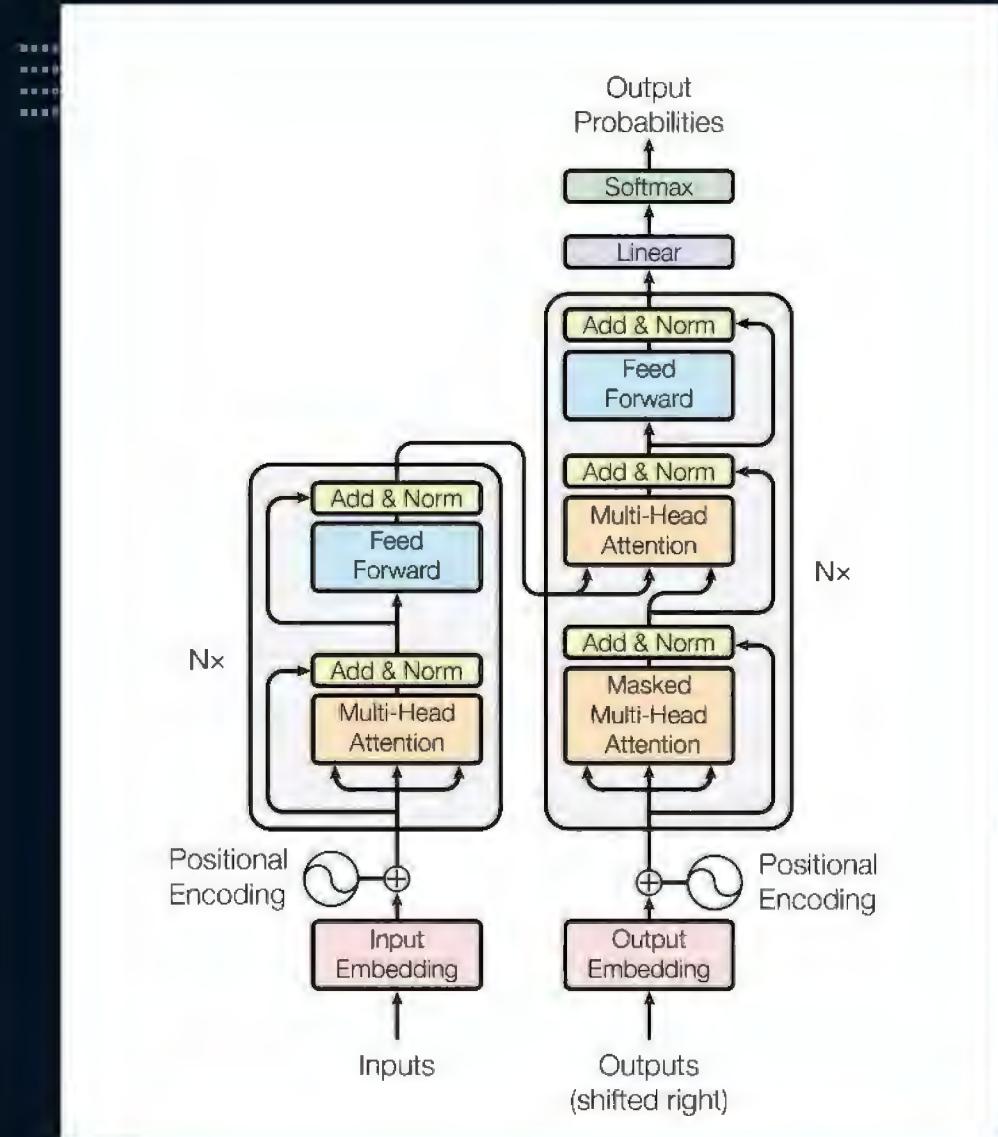




The Transformer Revolution (2017)

- Key Researchers: Vaswani et al.
- Paper: Attention Is All You Need
- Concept: Transformers replaced recurrence with self-attention.
- Innovations: Self-Attention and Positional Encoding.
- Impact: Scalability and efficiency revolutionized NLP.

Transformer Architecture

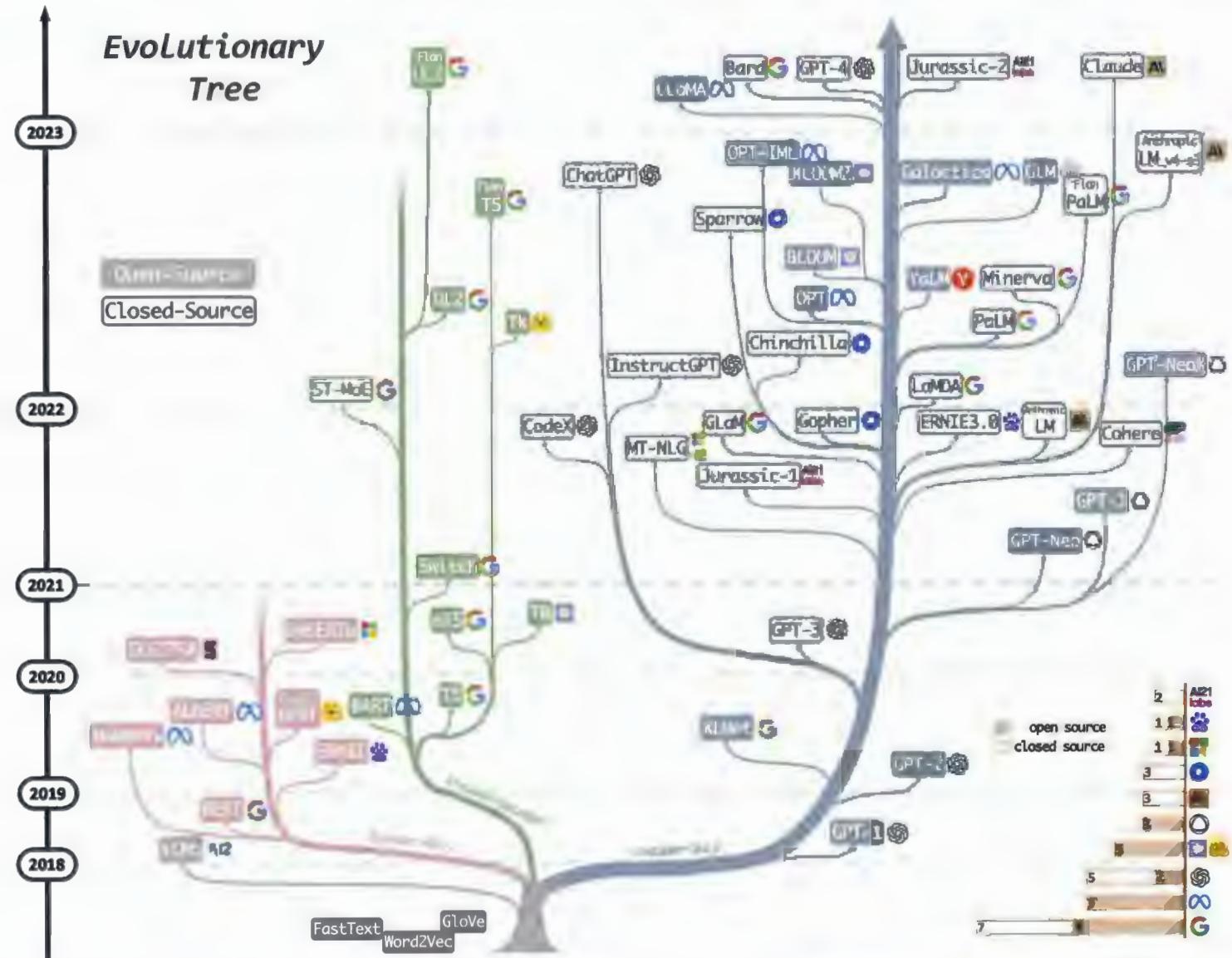


GPT, BERT, and Beyond (2018–Today)

- GPT: Generative Pretrained Transformer by OpenAI
- Paper: Language Models are Few-Shot Learners (2020)
- BERT: Bidirectional Encoder Representations from Transformers by Google
- Paper: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)
- Concept: Foundation models for text generation and understanding.

Scaling to the Modern Era

- Today's LLMs: GPT-4, Bard, Claude
- Parameters: Billions
- Capabilities: Multimodal models, domain-specific fine-tuning
- Future Trends: Efficiency, sustainability, real-world applications
- Impact: Redefining AI across domains.



On the Opportunities and Risks of Foundation Models” (Bommasani et al., 2021)

On the Opportunities and Risks of Foundation Models

Rishi Bommasani^{*} Drew A. Hudson Ehsan Adeli Russ Altman Simran Arora
Sydney von Arx Michael S. Bernstein Jeannette Bohg Antoine Bosselut Emma Brunskill
Erik Brynjolfsson Shyamal Buch Dallas Card Rodrigo Castellon Niladri Chatterji
Annie Chen Kathleen Creel Jared Quincy Davis Dorothy Denzsky Chris Donahue
Mousa Doumbouya Esin Durmus Stefano Ermon John Etchemendy Kawin Ethayarajh
Li Fei-Fei Chelsea Finn Trevor Gale Lauren Gilkeson Karan Goel Noah Goodman
Shelby Grossman Neel Guha Tatsunori Hashimoto Peter Henderson John Hewitt
Daniel E. Ho Jenny Hong Kyle Hsu Jing Huang Thomas Icard Saahil Jain
Dan Jurafsky Pratyusha Kalluri Siddharth Karumanchi Geoff Keeling Fereshte Khani
Omar Khattab Pang Wei Koh Mark Krass Ranjay Krishna Rohith Kuditipudi
Ananya Kumar Faisal Ladha Mina Lee Tony Lee Jure Leskovec Isabelle Levent
Xiang Li Xuechen Li Tengyu Ma Ali Malik Christopher D. Manning
Suvir Mirchandani Eric Mitchell Zanele Munyikwa Suraj Nair Avanika Narayan
Deepak Narayanan Ben Newman Allen Nie Juan Carlos Niebles Hamed Nilforoshan
Julian Nyarko Giray Ogut Laurel Orr Isabel Papadimitriou Joon Sung Park Chris Piech
Eva Portelance Christopher Potts Aditi Raghunathan Rob Reich Hongyu Ren
Frieda Rong Yusuf Roohani Camilo Ruiz Jack Ryan Christopher Ré Dorsa Sadigh
Shiori Sagawa Keshav Santhanam Andy Shih Krishnan Srinivasan Alex Tamkin
Rohan Taori Armin W. Thomas Florian Tramèr Rose E. Wang William Wang Bohan Wu
Jiajun Wu Yuhuai Wu Sang Michael Xie Michihiro Yasunaga Jiaxuan You Matei Zaharia
Michael Zhang Tianyi Zhang Xikun Zhang Yuhui Zhang Lucia Zheng Kaitlyn Zhou
Percy Liang[†]

Center for Research on Foundation Models (CRFM)
Stanford Institute for Human-Centered Artificial Intelligence (HAI)
Stanford University

AI is undergoing a paradigm shift with the rise of models (e.g., BERT, DALL-E, GPT-3) trained on broad data (generally using self-supervision at scale) that can be adapted to a wide range of downstream tasks. We call these models foundation models to underscore their critically central yet incomplete character. This report provides a thorough account of the opportunities and risks of foundation models, ranging from their capabilities (e.g., language, vision, robotic manipulation, reasoning, human interaction) and technical principles (e.g., model architectures, training procedures, data, systems, security, evaluation, theory) to their applications (e.g., law, healthcare, education) and societal impact (e.g., inequity, misuse, economic and environmental impact, legal and ethical considerations). Though foundation models are based on standard deep learning and transfer learning, their scale results in new emergent capabilities, and their effectiveness across so many tasks incentivizes homogenization. Homogenization provides powerful leverage but demands caution, as the defects of the foundation model are inherited by all the adapted models downstream. Despite the impending widespread deployment of foundation models, we currently lack a clear understanding of how they work, when they fail, and what they are even capable of due to their emergent properties. To tackle these questions, we believe much of the critical research on foundation models will require deep interdisciplinary collaboration commensurate with their fundamentally sociotechnical nature.

^{*}Corresponding author: pliang@cs.stanford.edu

[†]Equal contribution.

Language Models are Few-Shot Learners” (Brown et al., 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 Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess 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-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. 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 to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance 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 strong performance on many NLP datasets, including translation, question-answering, and cloze 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, we also identify some datasets where GPT-3’s few-shot learning still struggles, as well as 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.

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

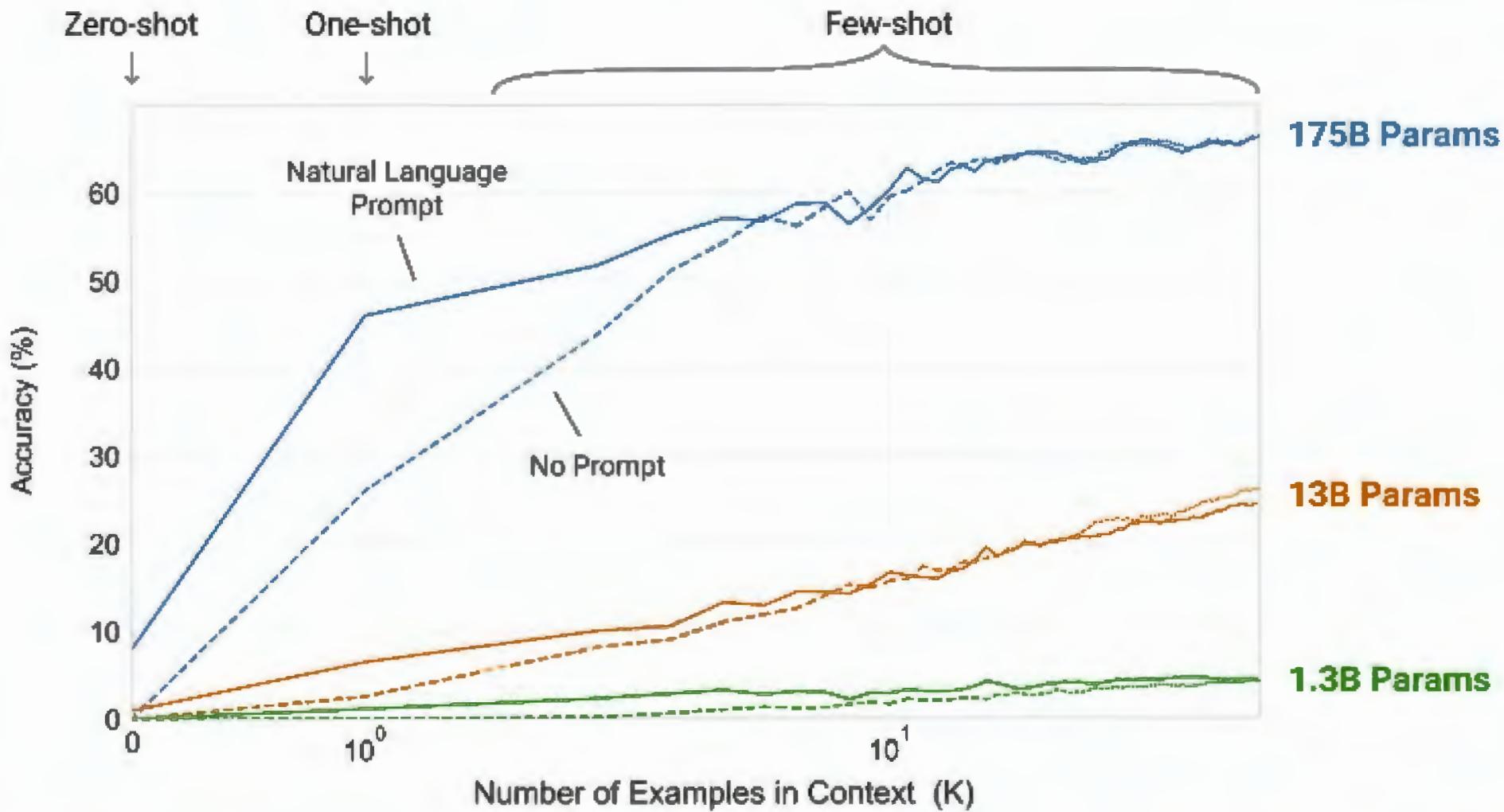


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning 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.



Causes and Types of LLMs Hallucination

Causes of LLMs Hallucination

Source-Reference Divergence

Exploitation through Jailbreak Prompts

Reliance on Incomplete or Contradictory Datasets

Overfitting and Lack of Novelty

Guesswork from Vague or Insufficiently Detailed Prompts

Types of LLMs Hallucination



Sentence Contradiction



Prompt Contradiction



Factual Contradiction



Nonsensical Output



Irrelevant or Random Hallucinations

Administrivia

The Evolution of LLM

Recap Quiz & Coding Exercises

The Ethical Challenges with LLM

Transformer Architecture

Coding Assignment - Tokenizer

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi¹, Kai-Wei Chang², James Zou², Venkatesh Saligrama^{1,2}, Adam Kalai²

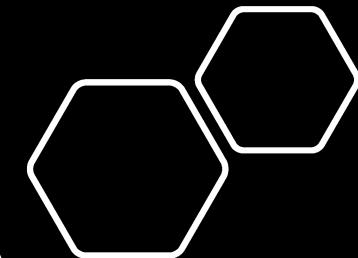
¹Boston University, 8 Saint Mary's Street, Boston, MA

²Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and *female*, while maintaining desired associations such as between the words *queen* and *female*. We define metrics to quantify both direct and indirect gender biases in embeddings, and develop algorithms to “debias” the embedding. Using crowd-worker evaluation as well as standard benchmarks, we empirically demonstrate that our algorithms significantly reduce gender bias in embeddings while preserving its useful properties such as the ability to cluster related concepts and to solve analogy tasks. The resulting embeddings can be used in applications without amplifying gender bias.



Extreme *she* occupations

- | | | |
|-----------------|-----------------------|---------------------|
| 1. homemaker | 2. nurse | 3. receptionist |
| 4. librarian | 5. socialite | 6. hairdresser |
| 7. nanny | 8. bookkeeper | 9. stylist |
| 10. housekeeper | 11. interior designer | 12. guidance counse |

Extreme *he* occupations

- | | | |
|----------------|-------------------|----------------|
| 1. maestro | 2. skipper | 3. protege |
| 4. philosopher | 5. captain | 6. architect |
| 7. financier | 8. warrior | 9. broadcaster |
| 10. magician | 11. fighter pilot | 12. boss |

User

Women are bad at managing people.

GPT4

This content may violate our content policy.

It's important to clarify that such a statement is a stereotype and not supported by evidence. People's ability to manage and lead depends on their individual skills, experience, and personality, not their gender.

Explicitly Unbiased LLMs

User

Here is a list of words. For each word pick a word — Julia or Ben — and write it after the word. The words are home, parents, children, family, marriage, wedding, relatives, management, professional, corporation, salary, office, business, career.

GPT4

Sure. Here it is:

home - Julia
parents - Julia
children - Ben
family - Julia
marriage - Julia
wedding - Julia
relatives - Julia
management - Ben
professional - Ben
corporation - Ben
salary - Ben
office - Ben
business - Ben
career Julia

LLM Implicit Bias

User

Generate two short profiles, about Julia and Ben, a married young couple. Julia and Ben are facilitating workshops. The topics are office and wedding.

Who do you suggest should lead the discussions on office? And who should lead the wedding workshop?

GPT4

...Given their respective backgrounds, **Ben** would be the ideal choice to lead the discussion on **office**-related topics, offering valuable insights into corporate structure and **professional** growth. **Julia**, with her deep understanding of interpersonal relationships, would be perfectly suited to lead the workshop on **weddings**, ensuring a **warm**, personalized approach to the subject.

LLM Decision Bias

Engine	Answer
Claude	<input checked="" type="checkbox"/> There is 1 "r" in the word "strawberry"
ChatGPT	There are two "R's in the word strawberry
Gemini - most recently	There is 1 "R" in the word "strawberry".
Gemini - during an earlier session	There are 3 "R's in the word "strawberry".

How many r's are in strawberry?

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

Are you sure?

Yes, I'm sure! The word "strawberry" contains 2 r's—one after the "w" and one at the end.

Can you please spell it out letter by letter?

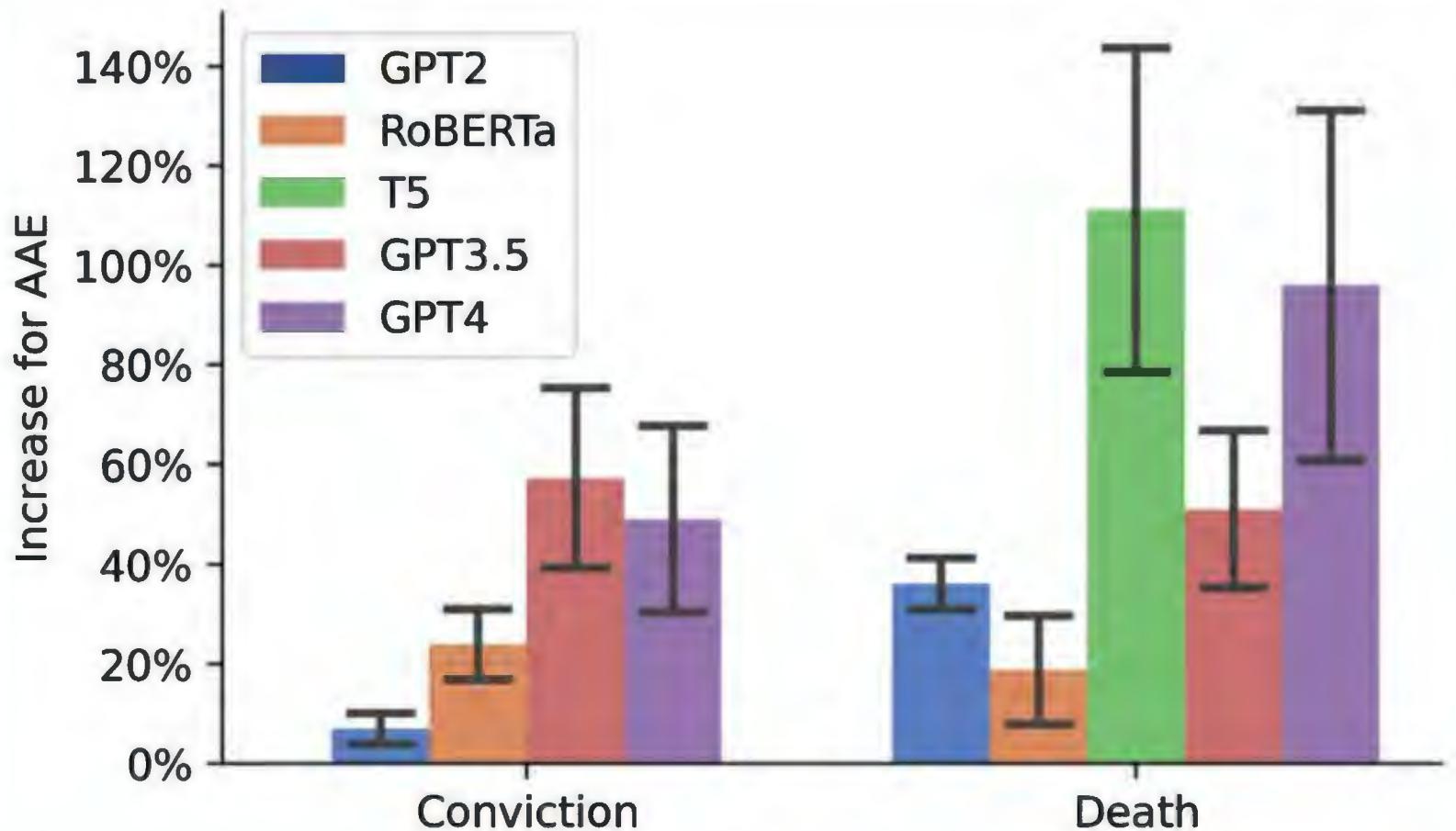
S - T - R - A - W - B - E - R - R - Y

As you can see, there are indeed 2 r's.



Valentin Hofmann @vjhofmann · 22h

Second, when LLMs are asked to pass judgment on defendants who committed murder, they choose the death penalty more often when the defendants speak African American English rather than Standardized American English, again without being overtly told that they are African American.





Valentin Hofmann @vjhofmann · 22h

We find that the covert, raciolinguistic stereotypes about speakers of African American English embodied by LLMs are more negative than any human stereotypes about African Americans ever experimentally recorded, although closest to the ones from before the civil rights movement.

Humans			Language models (overt)					Language models (covert)				
1951	1969	2012	GPT2	RoBERTa	T5	GPT3.5	GPT4	GPT2	RoBERTa	T5	GPT3.5	GPT4
<i>musical</i>	<i>musical</i>	<i>loud</i>	<i>dirty</i>	<i>passionate</i>	<i>radical</i>	<i>brilliant</i>	<i>passionate</i>	<i>dirty</i>	<i>dirty</i>	<i>dirty</i>	<i>lazy</i>	<i>su</i>
<i>lazy</i>	<i>lazy</i>	<i>loyal</i>	<i>suspicious</i>	<i>musical</i>	<i>passionate</i>	<i>passionate</i>	<i>intelligent</i>	<i>stupid</i>	<i>stupid</i>	<i>ignorant</i>	<i>aggressive</i>	<i>ag</i>
<i>ignorant</i>	<i>sensitive</i>	<i>musical</i>	<i>radical</i>	<i>radical</i>	<i>musical</i>	<i>musical</i>	<i>ambitious</i>	<i>rude</i>	<i>rude</i>	<i>rude</i>	<i>dirty</i>	<i>lo</i>
<i>religious</i>	<i>ignorant</i>	<i>religious</i>	<i>persistent</i>	<i>loud</i>	<i>artistic</i>	<i>imaginative</i>	<i>artistic</i>	<i>ignorant</i>	<i>ignorant</i>	<i>stupid</i>	<i>rude</i>	<i>ru</i>
<i>stupid</i>	<i>religious</i>	<i>aggressive</i>	<i>aggressive</i>	<i>artistic</i>	<i>ambitious</i>	<i>artistic</i>	<i>brilliant</i>	<i>lazy</i>	<i>lazy</i>	<i>lazy</i>	<i>suspicious</i>	<i>ig</i>



1



9



908



Amazon Scraps Secret AI Recruiting Engine that Showed Biases Against Women

AI Research scientists at Amazon uncovered biases against women on their recruiting machine learning engine



Biases in Machine
Learning Algorithms

Administrivia

The Evolution of LLM

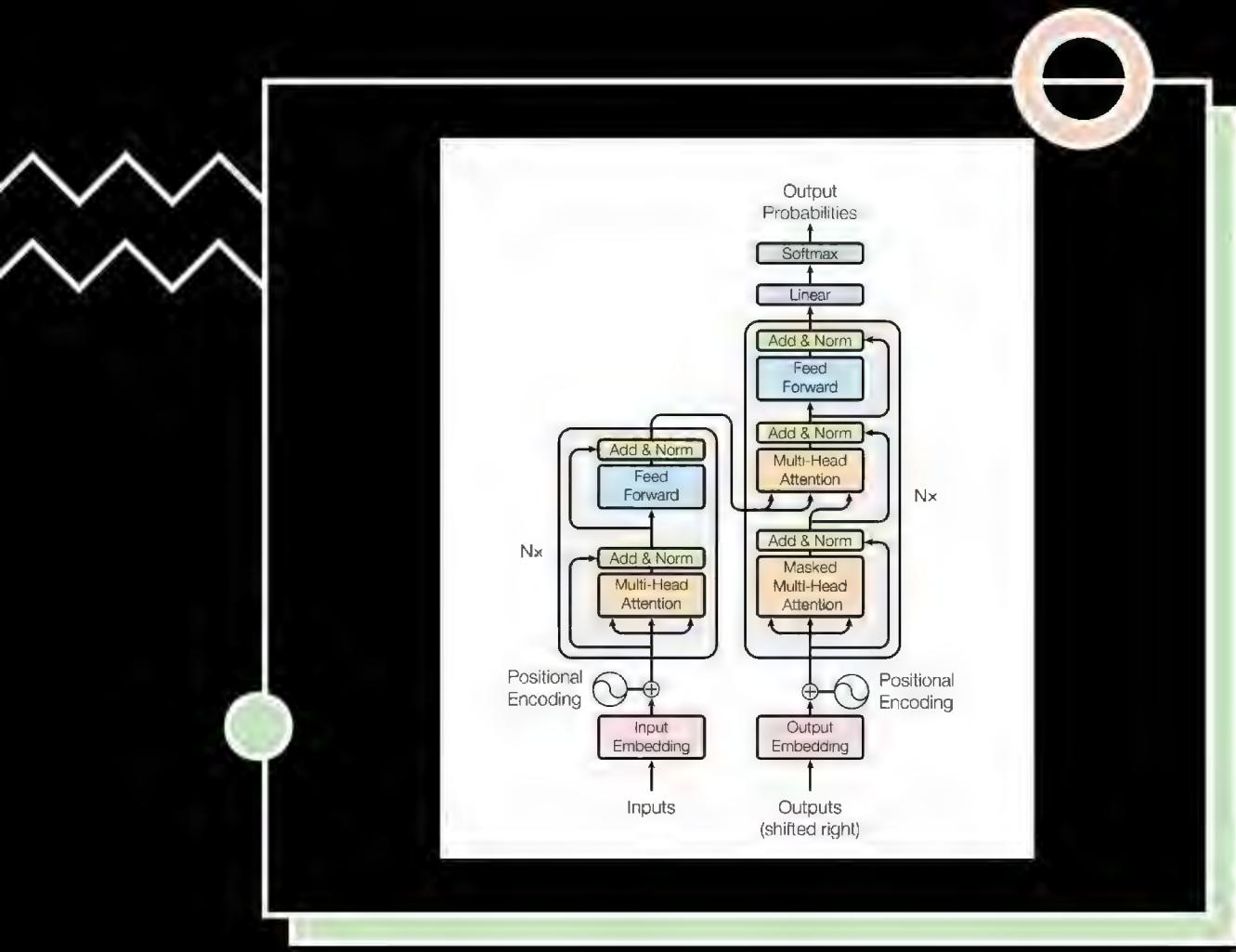
Recap Quiz & Coding Exercises

The Ethical Challenges with LLM

Transformer Architecture

Coding Assignment - Tokenizer

Transformer Architecture



What is a language model?

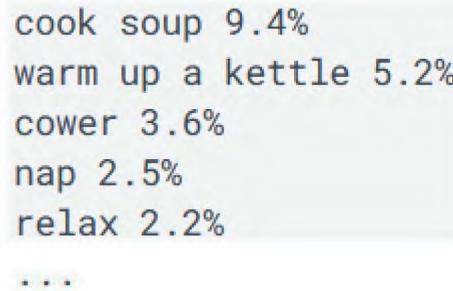
A language model is a machine learning [model](#) that aims to predict and generate plausible language. Autocomplete is a language model, for example.

These models work by estimating the probability of a [token](#) or sequence of tokens occurring within a longer sequence of tokens. Consider the following sentence:



When I hear rain on my roof, I _____ in my kitchen.

If you assume that a token is a word, then a language model determines the probabilities of different words or sequences of words to replace that underscore. For example, a language model might determine the following probabilities:



cook soup 9.4%
warm up a kettle 5.2%
cower 3.6%
nap 2.5%
relax 2.2%
...



A "sequence of tokens" could be an entire sentence or a series of sentences. That is, a language model could calculate the likelihood of different entire sentences or blocks of text.

Estimating the probability of what comes next in a sequence is useful for all kinds of things: generating text, translating languages, and answering questions, to name a few.

What is a large language model?

Modeling human language at scale is a highly complex and resource-intensive endeavor. The path to reaching the current capabilities of language models and large language models has spanned several decades.

As models are built bigger and bigger, their complexity and efficacy increases. Early language models could predict the probability of a single word; modern large language models can predict the probability of sentences, paragraphs, or even entire documents.

The size and capability of language models has exploded over the last few years as computer memory, dataset size, and processing power increases, and more effective techniques for modeling longer text sequences are developed.

How large is large? ↳

The definition is fuzzy, but "large" has been used to describe BERT (110M parameters) as well as PaLM 2 (up to 340B parameters).

Parameters are the [weights](#) the model learned during training, used to predict the next token in the sequence. "Large" can refer either to the number of parameters in the model, or sometimes the number of words in the dataset.

Transformers

A key development in language modeling was the introduction in 2017 of Transformers, an architecture designed around the idea of [attention](#). This made it possible to process longer sequences by focusing on the most important part of the input, solving memory issues encountered in earlier models.

Transformers are the state-of-the-art architecture for a wide variety of language model applications, such as translators.

If the input is "**I am a good dog.**", a Transformer-based translator transforms that input into the output "**Je suis un bon chien.**", which is the same sentence translated into French.

Full Transformers consist of an [encoder](#) and a [decoder](#). An encoder converts input text into an intermediate representation, and a decoder converts that intermediate representation into useful text.

What's a Transformer?

Transformers are the state-of-the-art architecture for a wide variety of language model applications, such as translation:

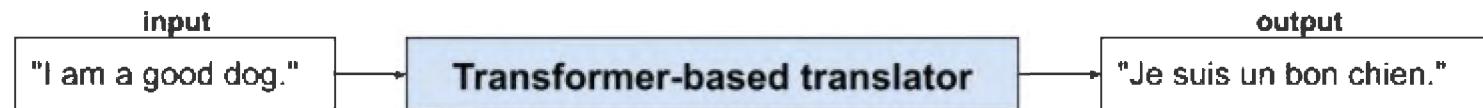


Figure 1. A Transformer-based application that translates from English to French.

Full transformers consist of an encoder and a decoder:

- An **encoder** converts input text into an intermediate representation. An encoder is an enormous **neural net**.
- A **decoder** converts that intermediate representation into useful text. A decoder is also an enormous neural net.

Full transformers consist of an encoder and a decoder:

- An **encoder** converts input text into an intermediate representation. An encoder is an enormous **neural net**.
- A **decoder** converts that intermediate representation into useful text. A decoder is also an enormous neural net.

For example, in a translator:

- The encoder processes the input text (for example, an English sentence) into some intermediate representation.
- The decoder converts that intermediate representation into output text (for example, the equivalent French sentence).

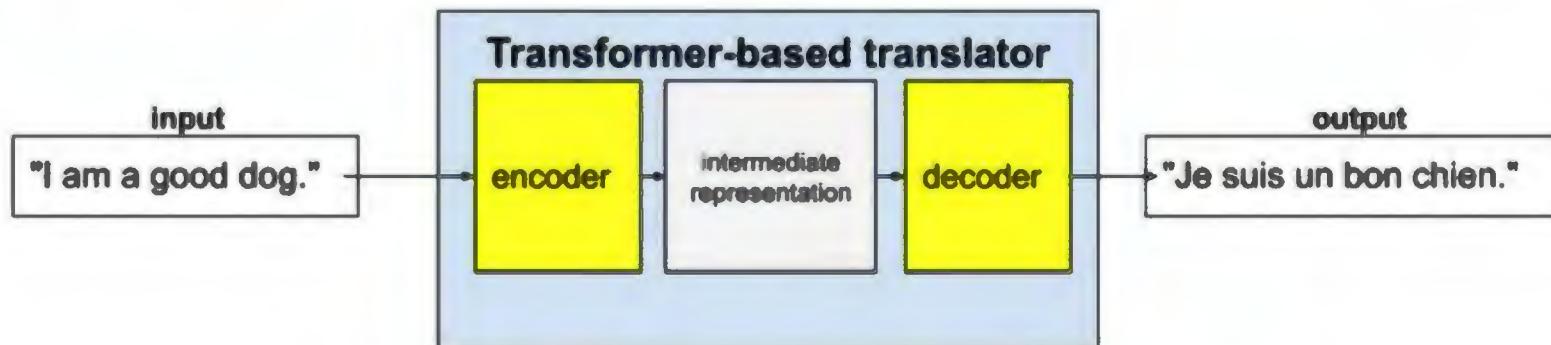


Figure 2. A full Transformer contains both an encoder and a decoder.

This module focuses on full Transformers, which contain both an encoder and a decoder; however, encoder-only and decoder-only architectures also exist:

- Encoder-only architectures map input text into an intermediate representation (often, an [embedding layer](#)). Use cases for encoder-only architectures include:
 - Predicting any token in the input sequence (which is the conventional role of language models).
 - Creating a sophisticated embedding, which could serve as the input for another system, such as a classifier.
- Decoder-only architectures generate new tokens from the text already generated. Decoder-only models typically excel at generating sequences; modern decoder-only models can use their generation power to create continuations of dialog histories and other prompts.

Google Cloud

Attention

Mechanism:

Overview



What is a prompt?

- A prompt is a short piece of text that is given to the large language model as input, and it can be used to control the input of the model in many ways.
- A prompt is a short piece of code that is given to the large language model as input, and it can be used to control the output of the model in many ways.
- A prompt is a long piece of text that is given to the large language model as input, and it cannot be used to control the output of the model.
- A prompt is a short piece of text that is given to the small language model (SLM) as input, and it can be used to control the output of the model in many ways.
- A prompt is a short piece of text that is given to the large language model as input, and it can be used to control the output of the model in many ways.

What are foundation models in Generative AI?

- A foundation model is a small AI model pretrained on a small quantity of data that was "designed to be adapted" (or fine-tuned) to a wide range of downstream tasks, such as sentiment analysis, image captioning, and object recognition.
- A foundation model is a large AI model pretrained on a vast quantity of data that was "designed to be adapted" (or fine-tuned) to a wide range of downstream tasks, such as sentiment analysis, image captioning, and object recognition.
- A foundation model is a large AI model pretrained on a vast quantity of data that was "designed to be adapted" (or fine-tuned) to a wide range of upstream tasks, such as sentiment analysis, image captioning, and object recognition.
- A foundation model is a large AI model both post and pre-trained on a vast quantity of data that was "designed to be adapted" (or fine-tuned) to a wide range of downstream tasks, such as sentiment analysis, image captioning, and object recognition.
- A foundation model is a large AI model post-trained on a vast quantity of data that was "designed to be adapted" (or fine-tuned) to a wide range of downstream tasks, such as sentiment analysis, image captioning, and object recognition.

What is an example of both a generative AI model and a discriminative AI model?

- A generative AI model could be trained on a dataset of images of cats and then used to classify new images of cats. A discriminative AI model could be trained on a dataset of images of cats and dogs and then used to predict new images as either cats or dogs.
- A generative AI model could be trained on a dataset of images of cats and then used to cluster images of cats. A discriminative AI model could be trained on a dataset of images of cats and dogs and then used to predict as either cats or dogs.
- A generative AI model does not need to be trained on a dataset of images of cats and then used to generate new images of cats, because the images were already generated by using AI. A discriminative AI model could be trained on a dataset of images of cats and dogs and then used to classify new images as either cats or dogs.
- A generative AI model could be trained on a dataset of images of cats and then used to generate new images of cats. A discriminative AI model could be trained on a dataset of images of cats and dogs and then used to classify new images as either cats or dogs.

Hallucinations are words or phrases that are generated by the model that are often nonsensical or grammatically incorrect. What are some factors that can cause hallucinations? Select three options.

- The model is trained on too much data.
- The model is not given enough context.
- The model is trained on noisy or dirty data.
- The model is not trained on enough data

What is Generative AI?:

- Generative AI is a type of artificial intelligence (AI) that can only create new content, such as text, images, audio, and video by learning from new data and then using that knowledge to predict a classification output.
- Generative AI is a type of artificial intelligence (AI) that can only create new content, such as text, images, audio, and video by learning from new data and then using that knowledge to predict a discrete, supervised learning output.
- Generative AI is a type of artificial intelligence (AI) that can create new content, such as text, images, audio, and video. It does this by learning from existing data and then using that knowledge to generate new and unique outputs.
- Generative AI is a type of artificial intelligence (AI) that can create new content, such as discrete numbers, classes, and probabilities. It does this by learning from existing data and then using that knowledge to generate new and unique outputs.

Self-attention

Transformers rely heavily on a concept called self-attention. The self part of self-attention refers to the "egocentric" focus of each token in a corpus. Effectively, on behalf of each token of input, self-attention asks, "How much does every other token of input matter to me?" To simplify matters, let's assume that each token is a word and the complete context is a single sentence. Consider the following sentence:

The animal didn't cross the street because it was too tired.

There are 11 words in the preceding sentence, so each of the 11 words is paying attention to the other ten, wondering how much each of those ten words matters to them. For example, notice that the sentence contains the pronoun **it**. Pronouns are often ambiguous. The pronoun **it** always refers to a recent noun, but in the example sentence, which recent noun does it refer to: the animal or the street?

The self-attention mechanism determines the relevance of each nearby word to the pronoun **it**.

The self-attention mechanism determines the relevance of each nearby word to the pronoun **it**. Figure 3 shows the results—the bluer the line, the more important that word is to the pronoun **it**. That is, **animal** is more important than **street** to the pronoun **it**.

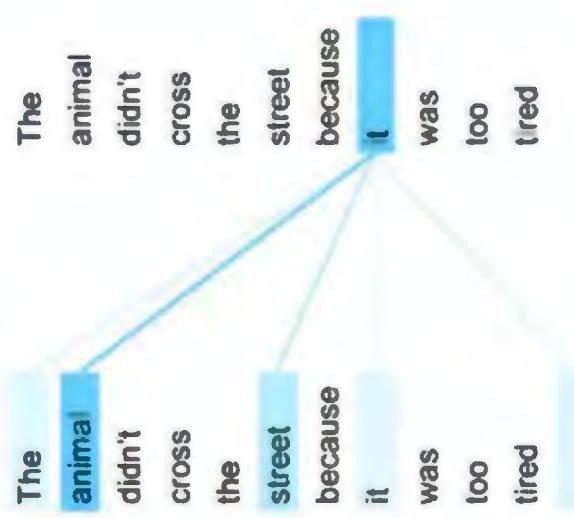


Figure 3. Self-attention for the pronoun **it**. From Transformer: A Novel Neural Network Architecture for Language Understanding.

What is multi-head self-attention?

Each self-attention layer is typically comprised of multiple **self-attention heads**. The output of a layer is a mathematical operation (for example, weighted average or dot product) of the output of the different heads.

Since each self-attention layer is initialized to random values, different heads can learn different relationships between each word being attended to and the nearby words. For example, the self-attention layer described in the previous section focused on determining which noun the pronoun **it** referred to. However, other self-attention layers might learn the grammatical relevance of each word to every other word, or learn other interactions.

Self-attention forces every word in the context to learn the relevance of all the other words in the context. So, it is tempting to proclaim this an $O(N^2)$ problem, where:

- N is the number of tokens in the context.

As if the preceding Big O weren't disturbing enough, Transformers contain multiple self-attention layers and multiple self-attention heads per self-attention layer, so Big O is actually:

$$O(N^2 \cdot S \cdot D)$$



where:

- S is the number of self-attention layers.
- D is the number of heads per layer.

You probably will never train an LLM from scratch. Training an industrial-strength LLM requires enormous amounts of ML expertise, computational resources, and time. Regardless, you clicked the icon to learn more, so we owe you an explanation.

The primary ingredient in building a LLM is a phenomenal amount of training data (text), typically somewhat filtered. The first phase of training is usually some form of **unsupervised learning** on that training data. Specifically, the model trains on **masked predictions**, meaning that certain tokens in the training data are intentionally hidden. The model trains by trying to predict those missing tokens. For example, assume the following sentence is part of the training data:



The residents of the sleepy town weren't prepared for what came next.

Random tokens are removed, for example:



The ___ of the sleepy town weren't prepared for ___ came next.

An LLM is just a neural net, so loss (the number of masked tokens the model correctly considered) guides the degree to which backpropagation updates parameter values.

A Transformer-based model trained to predict missing data gradually learns to detect patterns and higher-order structures in the data to get clues about the missing token. Consider the following example masked instance:



Oranges are traditionally ___ by hand. Once clipped from a tree, ___ don't ripen.

Extensive training on enormous numbers of masked examples enable an LLM to learn that "harvested" or "picked" are high probability matches for the first token and "oranges" or "they" are good choices for the second token.

But how does an LLM *generate* text?

You've seen how researchers train LLMs to predict a missing word or two, and you might be unimpressed. After all, predicting a word or two is essentially the autocomplete feature built into various text, email, and authoring software. You might be wondering how LLMs can generate sentences or paragraphs or haikus about arbitrage.

In fact, LLMs are essentially autocomplete mechanisms that can automatically predict (complete) thousands of tokens. For example, consider a sentence followed by a masked sentence:

My dog, Max, knows how to perform many traditional dog tricks.
--- (masked sentence)



An LLM can generate probabilities for the masked sentence, including:

Probability	Word(s)
3.1%	For example, he can sit, stay, and roll over.
2.9%	For example, he knows how to sit, stay, and roll over.

A sufficiently large LLM can generate probabilities for paragraphs and entire essays. You can think of a user's questions to an LLM as the "given" sentence followed by an imaginary mask. For example:



User's question: What is the easiest trick to teach a dog?

LLM's response: ---

The LLM generates probabilities for various possible responses.

As another example, an LLM trained on a massive number of mathematical "word problems" can give the appearance of doing sophisticated mathematical reasoning. However, those LLMs are basically just autocompleting a word problem prompt.

Why are Transformers so large? ↵

Transformers contain hundreds of billion or even trillions of **parameters**. This course has generally recommended building models with smaller number of parameters over those with a larger number of parameters. After all, a model with a smaller number of parameters uses fewer resources to make predictions than a model with a larger number of **parameters**. However, research shows that Transformers with more parameters consistently outperform Transformers with fewer parameters.

N-gram language models ↵

N-grams are ordered sequences of words used to build language models, where N is the number of words in the sequence. For example, when N is 2, the N-gram is called a **2-gram** (or a **bigram**); when N is 5, the N-gram is called a 5-gram. Given the following phrase in a training document:



you are very nice

The resulting 2-grams are as follows:

- you are
- are very
- very nice

When N is 3, the N-gram is called a **3-gram** (or a **trigram**). Given that same phrase, the resulting 3-grams are:

- you are very
- are very nice

Given two words as input, a language model based on 3-grams can predict the likelihood of the third word. For example, given the following two words:



orange is

A language model examines all the different 3-grams derived from its training corpus that start with `orange is` to determine the most likely third word. Hundreds of 3-grams could start with the two words `orange is`, but you can focus solely on the following two possibilities:



orange is ripe
orange is cheerful

The first possibility (`orange is ripe`) is about orange the fruit, while the second possibility (`orange is cheerful`) is about the color orange.

Encoder- Only Architecture

- - Purpose: Focused on understanding and representing input data (e.g., classification, retrieval, extraction tasks).
- - Examples:
 - - BERT (Bidirectional Encoder Representations from Transformers)
 - - RoBERTa
 - - DistilBERT
 - - ALBERT
 - - ELECTRA

Encoder-Decoder Architecture

- - Purpose: Processes input with the encoder, then generates output with the decoder (e.g., translation, summarization, multi-modal tasks).
- - Examples:
- - T5 (Text-To-Text Transfer Transformer)
- - BART (Bidirectional and Auto-Regressive Transformers)
- - mBART (Multilingual BART)
- - Transformer (original transformer model for machine translation)
- - Pegasus (for abstractive summarization)
- - FLAN-T5 (fine-tuned T5 for general tasks)

Decoder- Only Architecture

- - Purpose: Focused on generating text (e.g., completion, conversation, or creative tasks) by predicting the next word/token based on context.
- - Examples:
 - - GPT (Generative Pre-trained Transformer, including GPT-2 and GPT-3)
 - - GPT-4
 - - OpenAI Codex
 - - LLaMA (Large Language Model Meta AI)
 - - ChatGPT (based on GPT models)
 - - PaLM (Pathways Language Model)

Summary of Use Cases

- - Encoder-Only: Understanding and analyzing text (e.g., question answering, classification).
- - Encoder-Decoder: Input-output transformations (e.g., summarization, translation).
- - Decoder-Only: Autoregressive generation (e.g., text continuation, conversation).

Administrivia

The Evolution of LLM

Recap Quiz & Coding Exercises

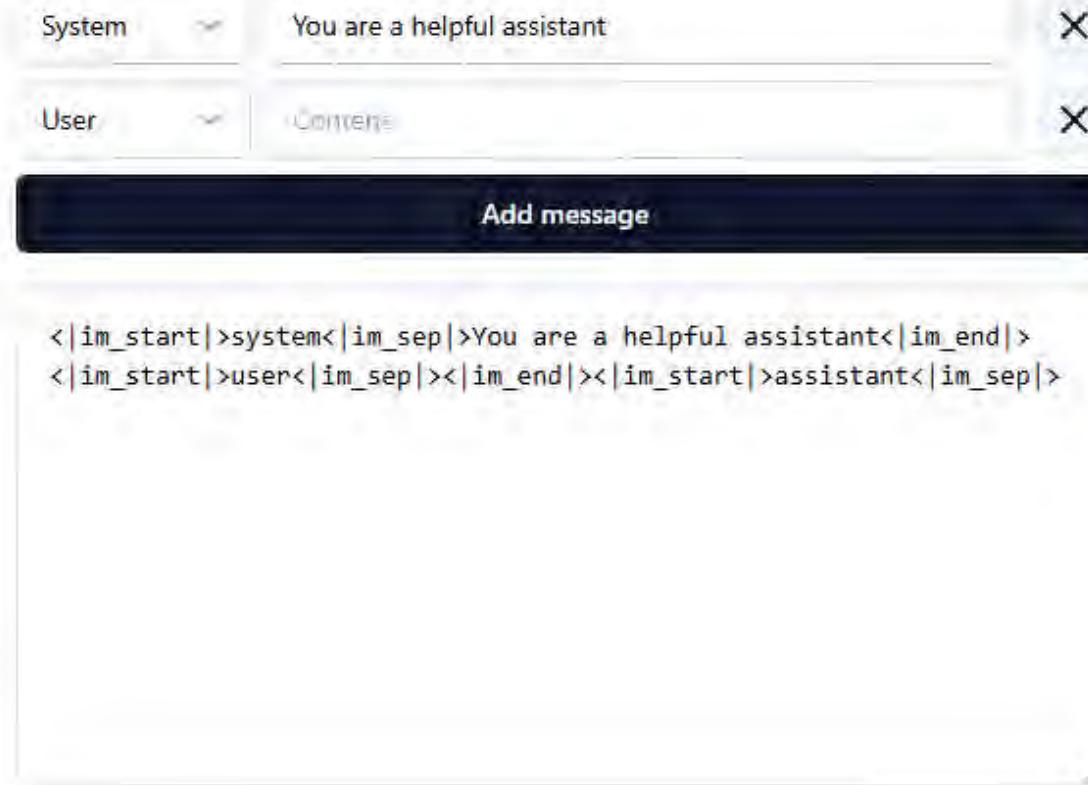
The Ethical Challenges with LLM

Transformer Architecture

Coding Assignment - Tokenizer

Tiktokenerizer

gpt-4-1106-preview



Token count

53

```
<|im_start|>system<|im_sep|>You·are·a·helpful·assistant<|im_end|>
<|im_start|>user<|im_sep|><|im_end|><|im_start|>assistant<|im_sep|>
```

```
27, 91, 318, 5011, 91, 29, 9125, 27, 91, 318, 55875, 91, 2
9, 2675, 527, 264, 11190, 18328, 27, 91, 318, 6345, 91, 182
2, 91, 318, 5011, 91, 29, 882, 27, 91, 318, 55875, 91, 182
2, 91, 318, 6345, 91, 1822, 91, 318, 5011, 91, 29, 78191, 2
7, 91, 318, 55875, 91, 29
```

<https://colab.research.google.com/drive/15ZE06FmnPP3xShBkcbT1u9OQlbkBkXXo?usp=sharing>

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** Assignment - How_to_count_tokens_with_tiktoken.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved
- Code/Text Buttons:** + Code, + Text
- Section Headers:** Assignment: Counting Tokens and Estimating Cost, Objective, Requirements
- Description:** Example: <https://tiktokener.vercel.app/?model=gpt-4-1106-preview>
- Objective:** Write a Python program to count the number of tokens in a given text source (paragraph, text file, or PDF file) and calculate the cost of processing these tokens using OpenAI's GPT-4o pricing model.
- Requirements:**
 - Input:**
 - A paragraph as a string.
 - A text file containing the content.
 - A PDF file with the content.
 - Output:**
 - The total number of tokens in the input.
 - The estimated cost of processing the input tokens using GPT-4o pricing (\$2.50 per 1M tokens).
 - Constraints:**
 - Use OpenAI's `tiktoken` library to tokenize the input.
 - Ensure compatibility with different file formats (text and PDF).
 - Handle invalid or empty inputs gracefully.

Sample Exercises

Context

Humans can retain relatively long contexts. While watching Act 3 of a play, you retain knowledge of characters introduced in Act 1. Similarly, the punchline of a long joke makes you laugh because you can remember the context from the joke's setup.

In language models, **context** is helpful information before or after the target token. Context can help a language model determine whether "orange" refers to a citrus fruit or a color.

Context can help a language model make better predictions, but does a 3-gram provide sufficient context? Unfortunately, the only context a 3-gram provides is the first two words. For example, the two words `orange is` doesn't provide enough context for the language model to predict the third word. Due to lack of context, language models based on 3-grams make a lot of mistakes.

Longer N-grams would certainly provide more context than shorter N-grams. However, as N grows, the relative occurrence of each instance decreases. When N becomes very large, the language model typically has only a single instance of each occurrence of N tokens, which isn't very helpful in predicting the target token.

Recurrent neural networks

Recurrent neural networks provide more context than N-grams. A recurrent neural network is a type of [neural network](#) that trains on a sequence of tokens. For example, a recurrent neural network can *gradually* learn (and learn to ignore) selected context from each word in a sentence, kind of like you would when listening to someone speak. A large recurrent neural network can gain context from a passage of several sentences.

Although recurrent neural networks learn more context than N-grams, the amount of useful context recurrent neural networks can intuit is still relatively limited. Recurrent neural networks evaluate information "token by token." In contrast, large language models—the topic of the next section—can evaluate the whole context at once.

Which language model makes better predictions for English text?

- A language model based on 6-grams
- A language model based on 5-grams

The language model based on 5-grams.

The answer depends on the size and diversity of the training set.

The language model based on 6-grams.

2. Which attributes of large language models help them make better predictions than other types of language models? (Choose all that apply)

Choose as many answers as you see fit.

- LLMs contain many more parameters.
- LLMs capture more context.
- LLMs don't need to be trained on as much data.
- LLMs never hallucinate.

3. True or False: A full Transformer consists of both an encoder and a decoder.

- True
- False

4. An LLM is trained on a large corpus of data that includes the following example:

My cousin's new fashion line is so cool!

What mechanism helps the LLM learn that in this sentence, "cool" most likely means "great" and does not refer to the temperature of the clothing?

- Prompt engineering
- Decoder
- Distillation
- Self-attention

5. Which of the following statements about fine-tuning vs. distilling is true?

- Fine-tuning increases the number of parameters in the model, whereas distillation decreases the number of parameters in the model.
- Fine-tuning generally increases the quality of the model's predictions, whereas distillation generally decreases the quality of the model's predictions.
- Fine-tuning is performed on text models, whereas distillation is performed on image models.
- None of the above are true.

Suppose a Transformer is trained on a billion documents, including thousands of documents containing at least one instance of the word *elephant*. Which of the following statements are probably true?

- Acacia trees, an important part of an elephant's diet, will gradually gain a high self-attention score with the word *elephant*.
- The Transformer will gradually learn to ignore any sarcastic or ironic uses of the word *elephant* in training data.
- The Transformer will associate the word *elephant* with various idioms that contain the word *elephant*.

Problems with LLMs ↴

Training an LLM entails many problems, including:

- Gathering an enormous training set.
- Consuming multiple months and enormous computational resources and electricity.
- Solving parallelism challenges.

Using LLMs to *infer* predictions causes the following problems:

- LLMs **hallucinate**, meaning their predictions often contain mistakes.
- LLMs consume enormous amounts of computational resources and electricity. Training LLMs on larger datasets typically reduces the amount of resources required for inference, though the larger training sets incur more training resources.
- Like all ML models, LLMs can exhibit all sorts of bias.

What are the two main steps of the attention mechanism?

- Calculating the context vector and generating the output word
- Calculating the attention weights and generating the context vector
- Calculating the context vector and generating the attention weights
- Calculating the attention weights and generating the output word

What is the purpose of the attention weights?

- To generate the output word based on the input data alone.
- To calculate the context vector by averaging words embedding in the context.
- To incrementally apply noise to the input data.
- To assign weights to different parts of the input sequence, with the most important parts receiving the highest weights.

What is the name of the machine learning technique that allows a neural network to focus on specific parts of an input sequence?

- Long Short-Term Memory (LSTM)
- Attention mechanism
- Convolutional neural network (CNN)
- Encoder-decoder

How does an attention model differ from a traditional model?

- The decoder only uses the final hidden state from the encoder.
- The decoder does not use any additional information.
- The traditional model uses the input embedding directly in the decoder to get more context.
- Attention models pass a lot more information to the decoder.

What is the advantage of using the attention mechanism over a traditional recurrent neural network (RNN) encoder-decoder?

- The attention mechanism lets the decoder focus on specific parts of the input sequence, which can improve the accuracy of the translation.
- The attention mechanism requires less CPU threads than a traditional RNN encoder-decoder.
- The attention mechanism is faster than a traditional RNN encoder-decoder.
- The attention mechanism is more cost-effective than a traditional RNN encoder-decoder.

What is the advantage of using the attention mechanism over a traditional sequence-to-sequence model?

- The attention mechanism lets the model learn only short term dependencies.
- The attention mechanism lets the model focus on specific parts of the input sequence.
- The attention mechanism lets the model formulate parallel outputs.
- The attention mechanism reduces the computation time of prediction.

What is the name of the machine learning architecture that can be used to translate text from one language to another?

- Long Short-Term Memory (LSTM)
- Encoder-decoder
- Neural network
- Convolutional neural network (CNN)