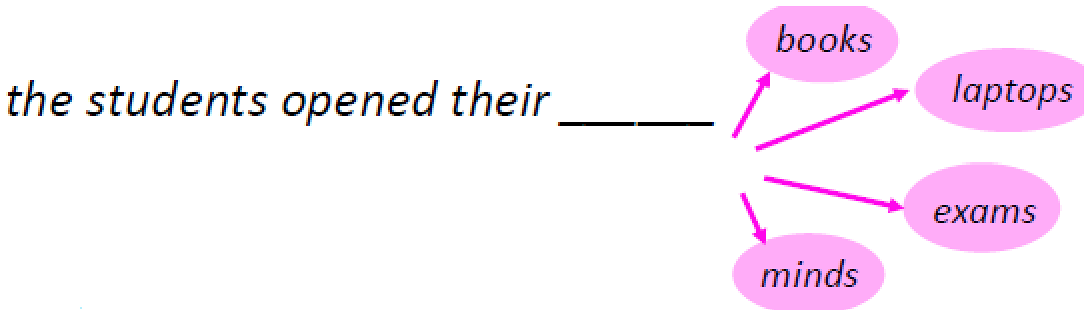
Introduction to Large Language Models

Language Modeling

- Language Modeling is the task of predicting what word comes next or the probability of a sentence.



- Goal:
  – compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

  – compute probability of an upcoming word:

$$P(w_5 \mid w_1,w_2,w_3,w_4)$$

- A system that does this is called a Language Model (LM).

What can you do with next-word prediction?

- A sufficiently strong (!) language model can do many, many things

Stanford University is located in __________, California. [Trivia]

I put ___ fork down on the table. [syntax]

The woman walked across the street, checking for traffic over ___ shoulder. [coreference]

I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___. [sentiment]

Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the ______. [some reasoning – this is harder]

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____ [some basic arithmetic]

Source: CS224N

Why word prediction?

It's how **large language models (LLMs)** work!

LLMs are **trained** to predict words

- Left-to-right (autoregressive) LMs learn to predict next word

LLMs **generate** text by predicting words

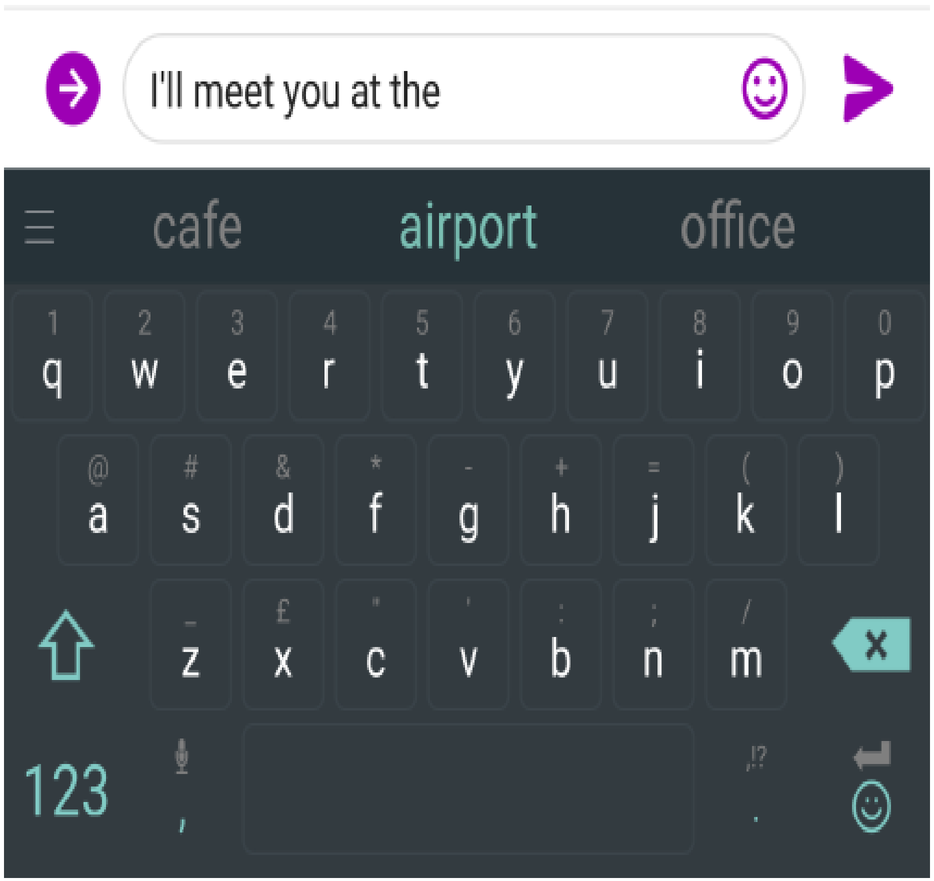
- By predicting the next word over and over again

Applications

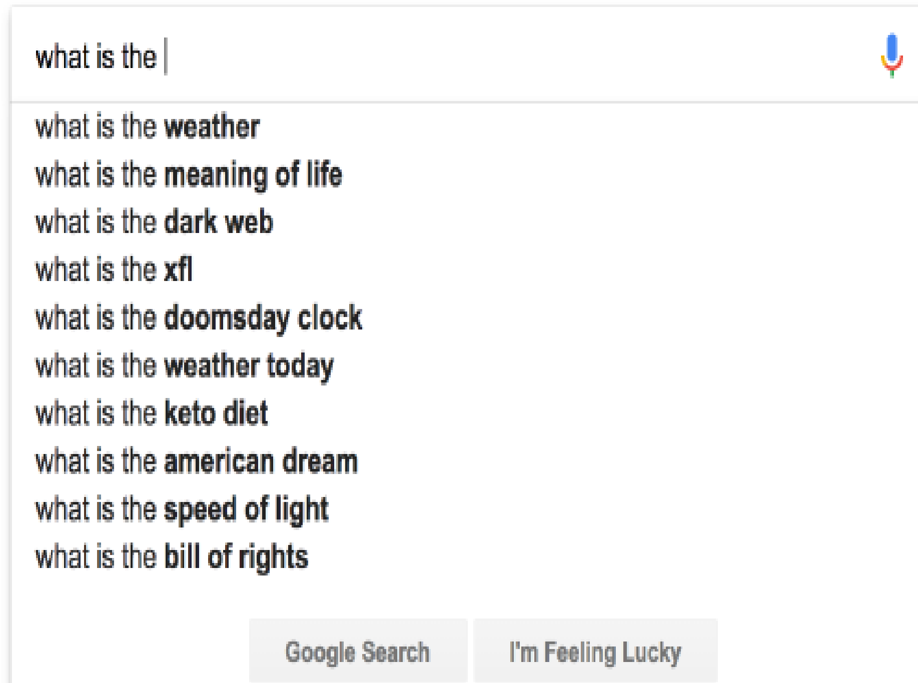
- Language Modeling is a subcomponent of many NLP tasks, especially those involving generating text or estimating the probability of text:
  - **Predictive typing**
  - **Speech recognition**
  - **Handwriting recognition**
  - **Spelling/grammar correction**
  - **Authorship identification**
  - **Machine translation**
  - **Summarization**
  - **Dialogue**
  - **etc.**

- Many tasks in NLP has been rebuilt upon Language Modeling.
- ChatGPT is an LM!

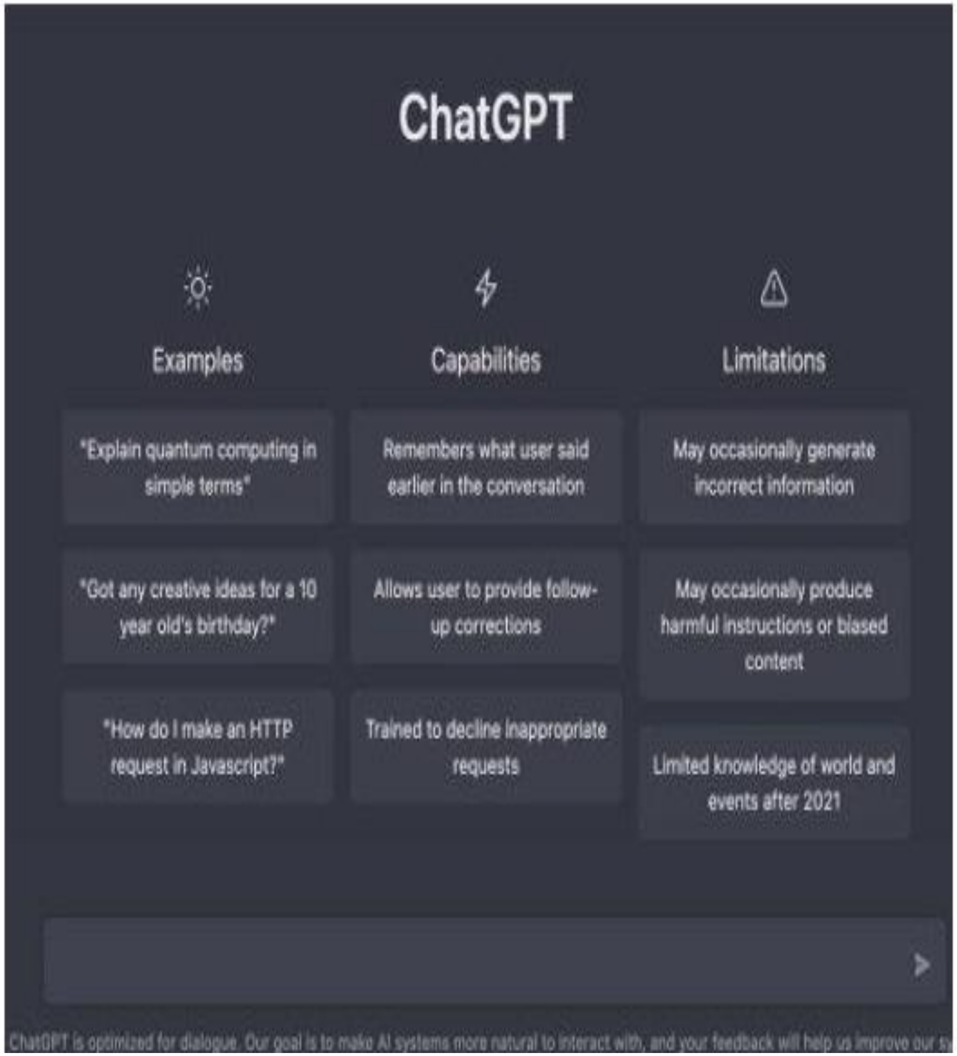
We use Language Models every day!

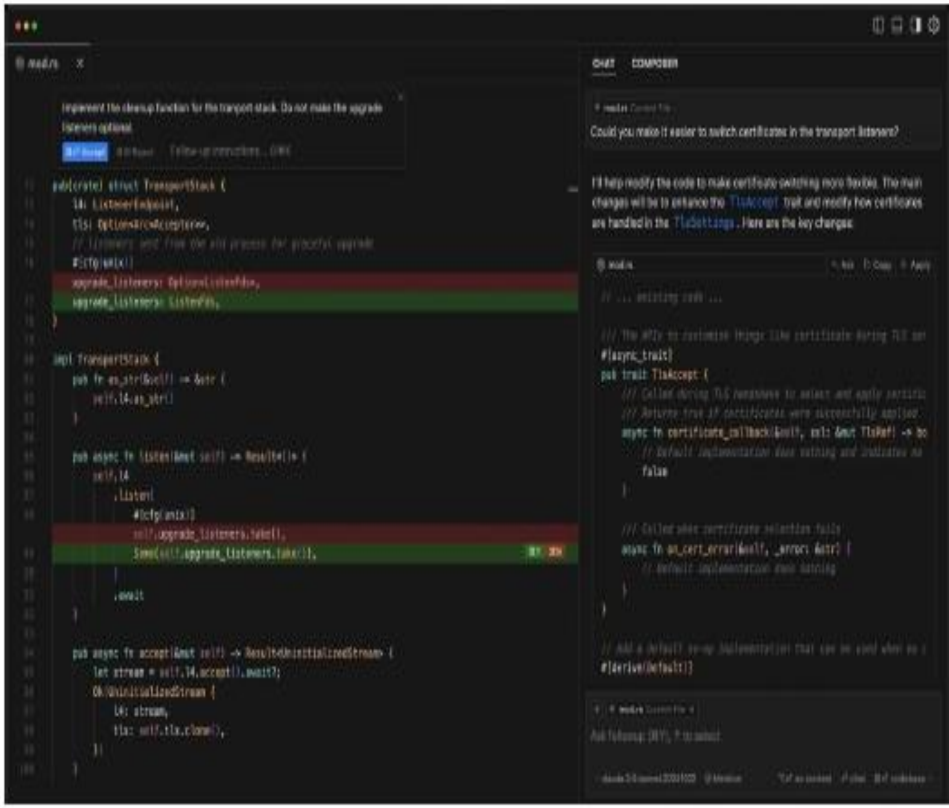


Google

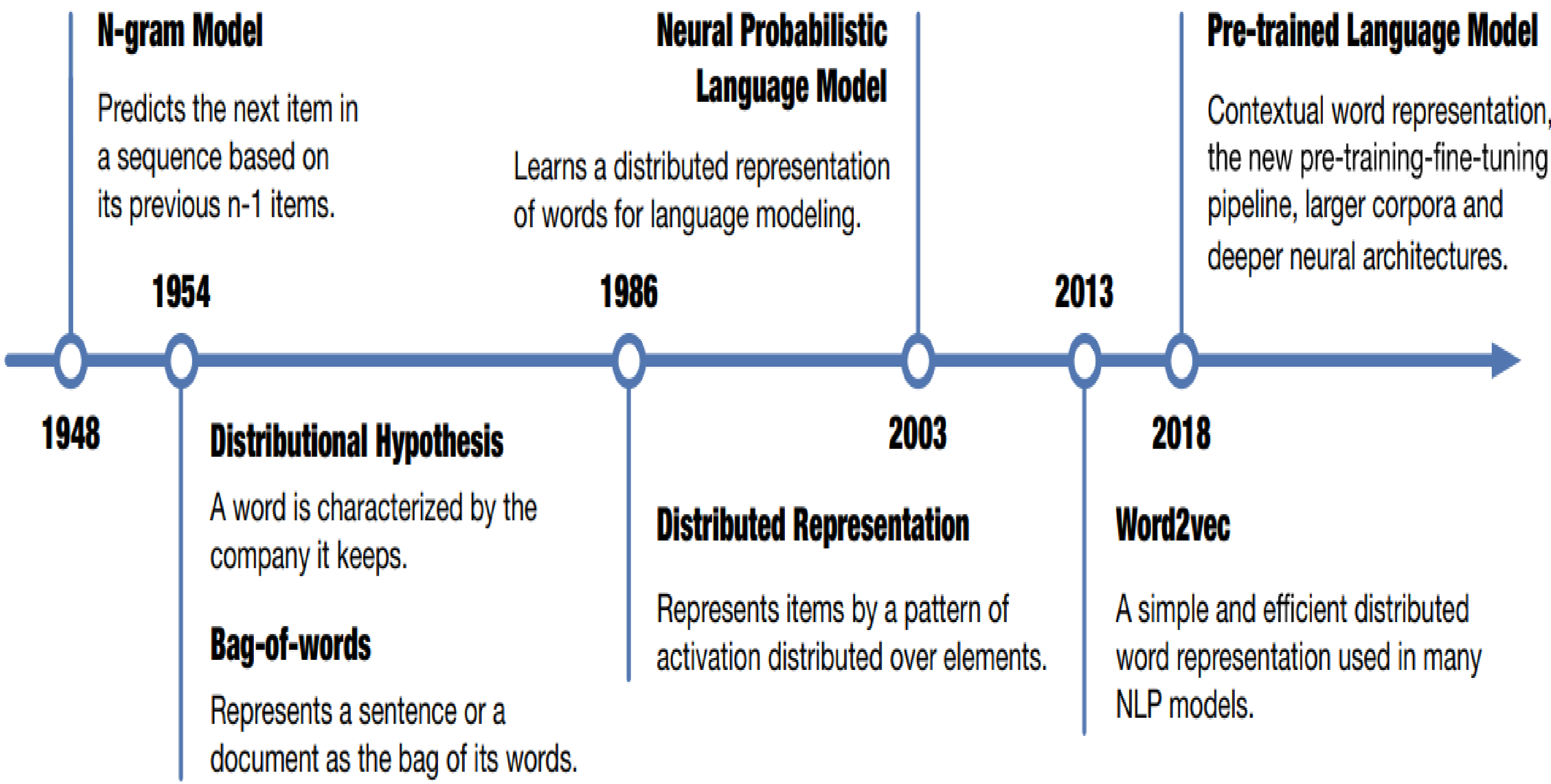


We use Language Models every day!





Developments in Representation Learning and LMs



- With the growing computing power and large-scale text data, distributed representation trained with neural networks and large corpora has become the mainstream.

Source: Liu et al., Representation Learning for Natural Language Processing, Springer, 2020

n-gram Language Models

Question: How to learn a Language Model?

Answer (pre- Deep Learning): learn an *n*-gram Language Model!

- The simplest model that assigns probabilities to sentences and sequences of words, *the n-gram Language Model*.
- An n-gram is a sequence of n consecutive words:
  - unigrams: “the”, “students”, “opened”, ”their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - four-grams: “the students opened their”
- **How to estimate probabilities?**
  - **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

An example for bi-gram

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

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

n-gram LM with Shakespeare Corpus

- Shakespeare Corpus contains the complete works, plays, sonnets, and poems of Shakespeare.
- N=884,647 tokens, V=29,066
- The next slide shows random sentences generated from unigram, bigram, trigram, and 4-gram models trained on Shakespeare's works.
- The longer the context on which we train the model, the more coherent the sentences. In the unigram sentences, there is no coherent relation between words or any sentence-final punctuation.
- The trigram and 4-gram sentences are beginning to look a lot like Shakespeare.
- Indeed, a careful investigation of the 4-gram sentences shows that they look a little too much like Shakespeare.
- The words "*It cannot be but so*" are directly from King John.

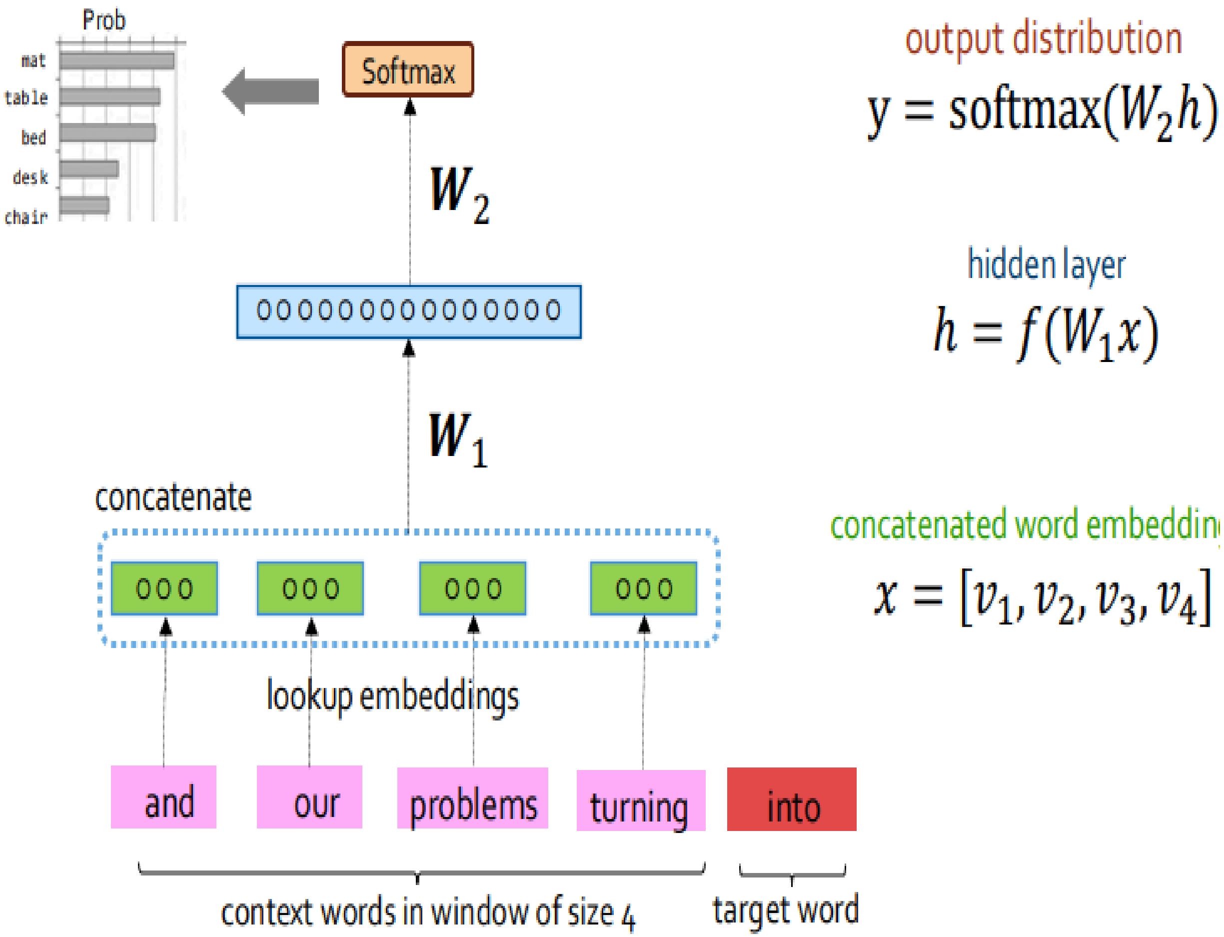
Approximating Shakespeare

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.

Language Models: History

- Probabilistic n-gram models of text generation [Jelinek+ 1980’s, …]
  - Applications: Speech Recognition, Machine Translation
- Statistical or shallow neural LMs (late 90’s – mid 00’s) [Bengio+ 2001, …]
- Recurrent neural nets (2010s)
- Pre-training deep neural language models (2017’s onward):
  - Many models based on: Self-Attention

Neural Network (NN)- Language Model



[Bengio et al. 2003]

Improvements over n-gram LM:

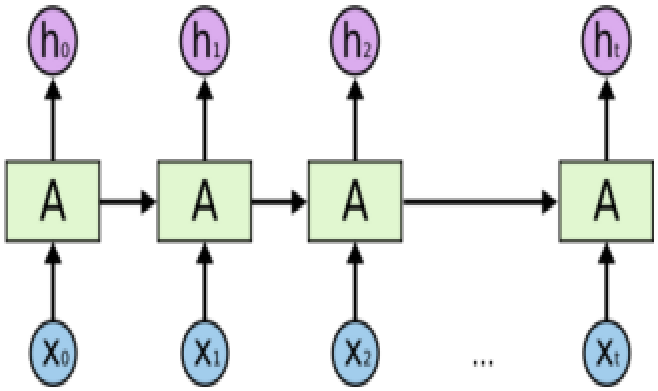
- Tackles the sparsity problem
- Model size is O(n) not O(exp(n)) — n being the window size.

Remaining problems:

- Fixed window is too small
- Enlarging window enlarges ***W*** — Window can never be large enough!
- It's not deep enough to capture nuanced contextual meanings

https://self-supervised.cs.jhu.edu/sp2025/files/slides/07.mlp-language-modeling.pdf

Recurrent Neural Network(RNN) Language Models

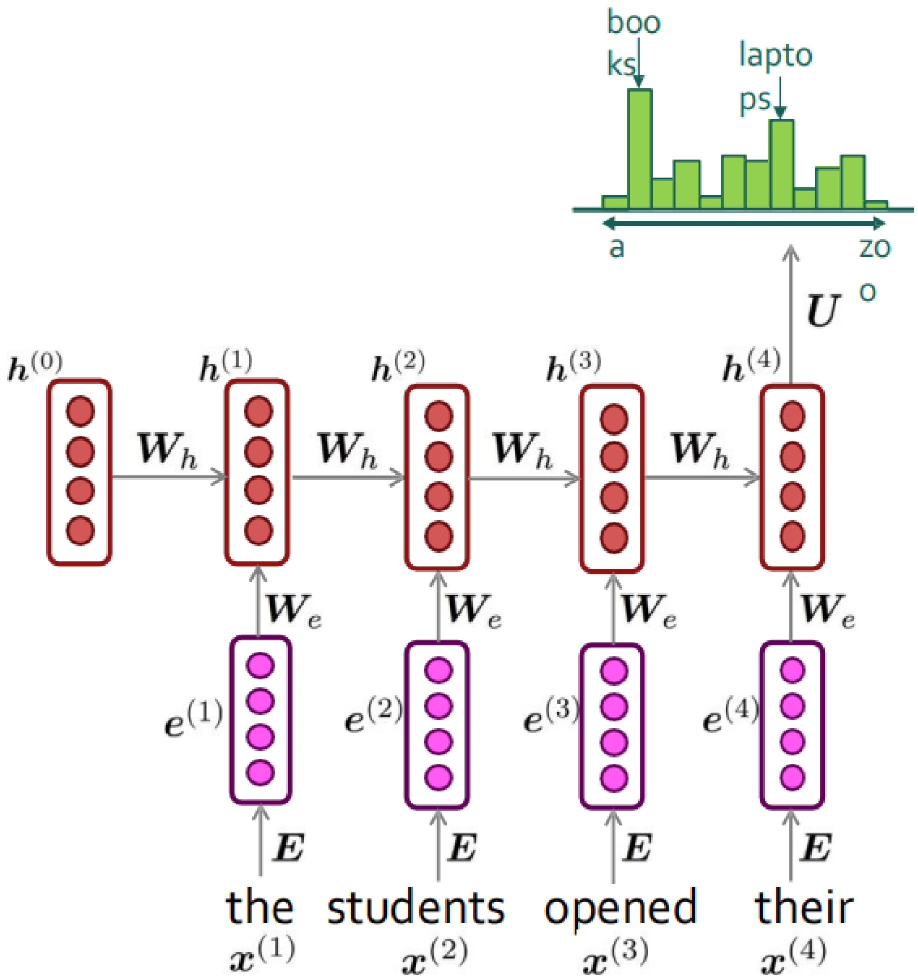


$$P(\underbrace{X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

- We feed the words one at a time to the RNN.
- A predictive head uses the latest embedding vector to produce a probability over the vocabulary.

RNNs: Weaknesses

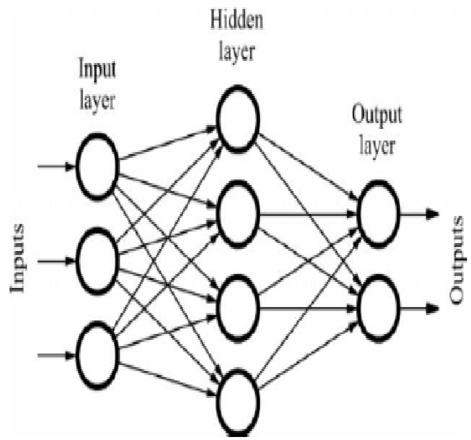
- Recurrent computation is slow and difficult to parallelize.
  - self-attention mechanism, better at representing long sequences and also parallelizable.
- While RNNs in theory can represent long sequences, they quickly forget portions of the input.
- Hard to learn long-distance dependencies



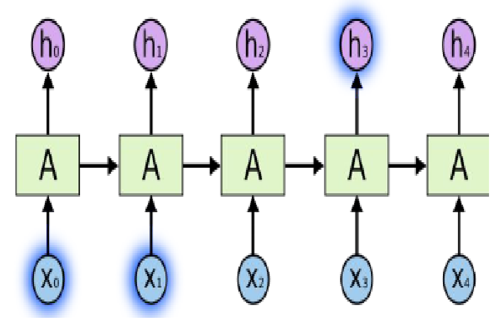
https://self-supervised.cs.jhu.edu/sp2025/files/slides/08-9.recurrent-nets.pdf

Which neural networks should be used for LLM?

- ✔ Multilayer Perceptron (MLP)
- ✔ Convolutional neural network
- ✔ Recurrent neural network
- ✔ **Transformer**

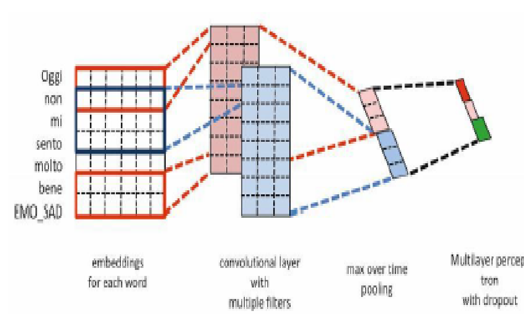


MLP

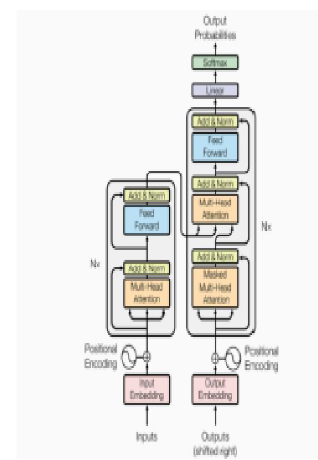


Recurrent NNs

Convolutional NNs



Transformer



Source: CSC6203

Which neural networks should be used for LLM?

- MLP

  +: Strongest inductive bias: if all words are concatenated

  +: Weakest inductive bias: if all words are averaged

  - : The interaction at the token-level is too weak

- CNN & RNN

  +: The interaction at the token-level is slightly better.

  CNN: Bringing the global token-level interaction to the window-level

  - : Make simplifications, its global dependencies are limited

  RNN: An ideal method for processing token sequences

  - : Its recursive nature has the problem of disaster forgetting.

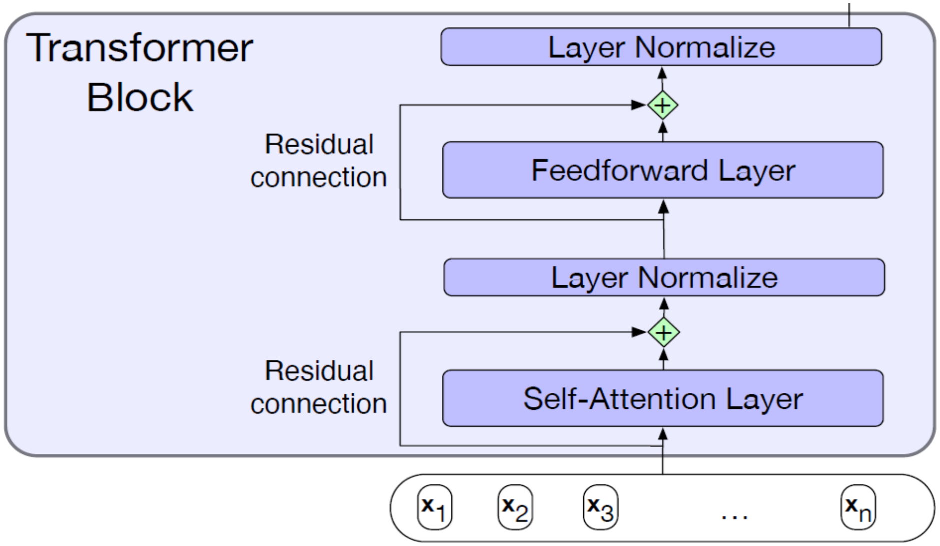
- **Transformer**

  +: Achieve **global dependence** at the **token-level** by **decoupling** token-level interaction and feature-level abstraction into two components, in **SAN** and **FNN**.

Source: CSC6203

Transformers

- Transformers map sequences of input vectors ($x_1,\ldots,x_n$) to sequences of output vectors ($y_1,\ldots,y_n$) of the same length.
- Transformers are made up of stacks of transformer blocks, each of which is a multilayer network made by combining simple linear layers, feedforward networks, and **attention layers**, the key innovation of transformers.
- Self-attention allows a network to directly extract and use information from arbitrarily large contexts without the need to pass it through intermediate recurrent connections as in RNNs.



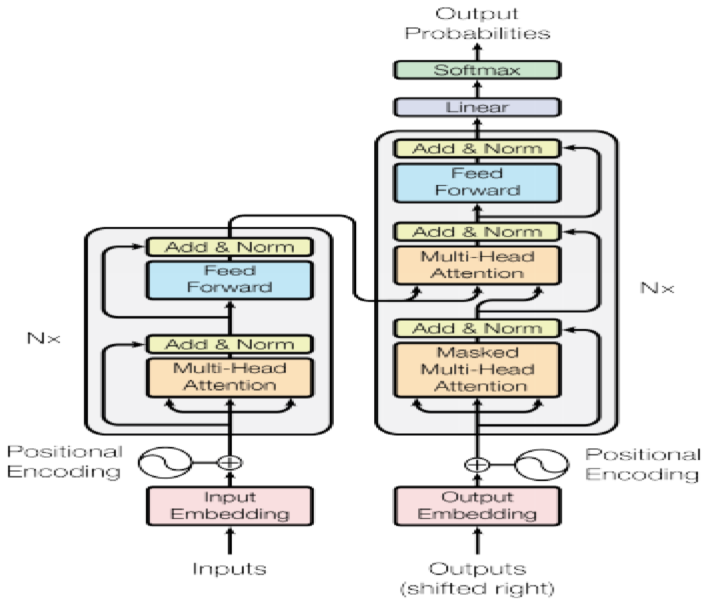


Figure 1: The Transformer - model architecture.

Transformers as Language Models

- Fig. 10.7 illustrates the general training approach. At each step, given all the preceding words, the final transformer layer produces an output distribution over the entire vocabulary. During training, the probability assigned to the correct word is used to calculate the cross-entropy loss for each item in the sequence

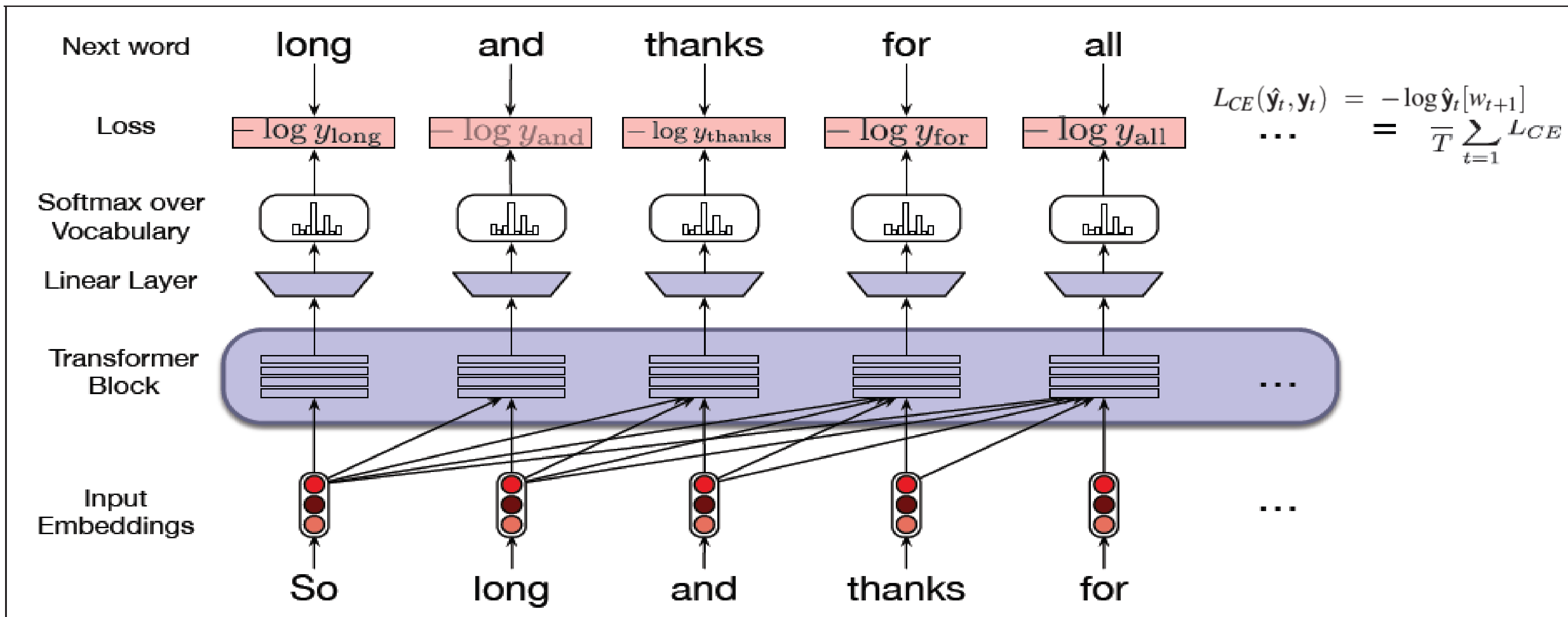
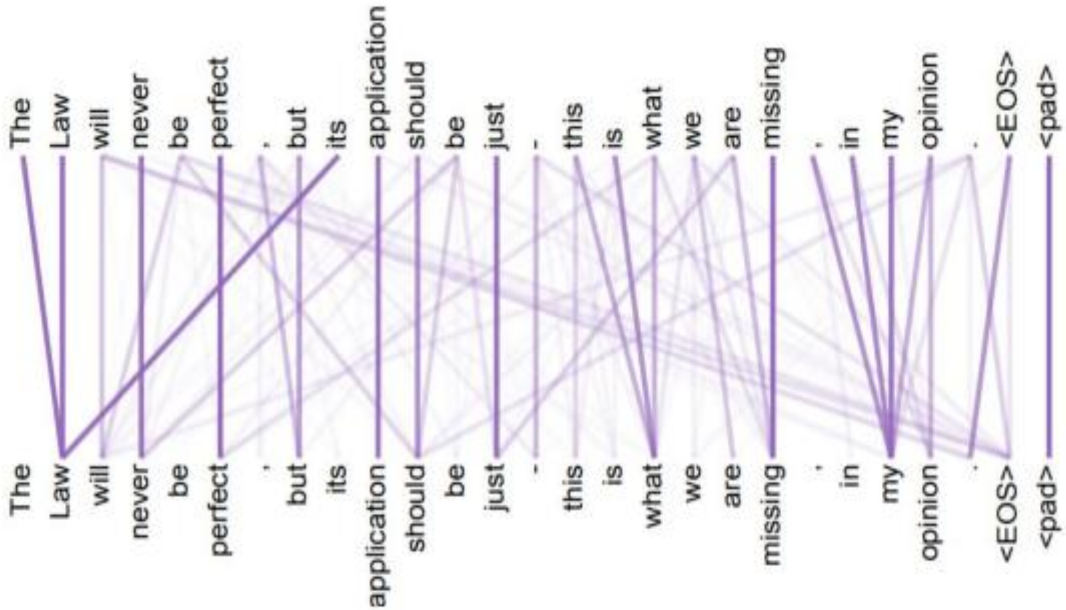


Figure 10.7 Training a transformer as a language model.

Attention

- Core idea: build a mechanism to focus ("attend") on a particular part of the context.
- How can this overcome the "long-range dependencies" problem in RNNs? By allow the model to **directly "look"** 👀 at all tokens and decide which one is useful.

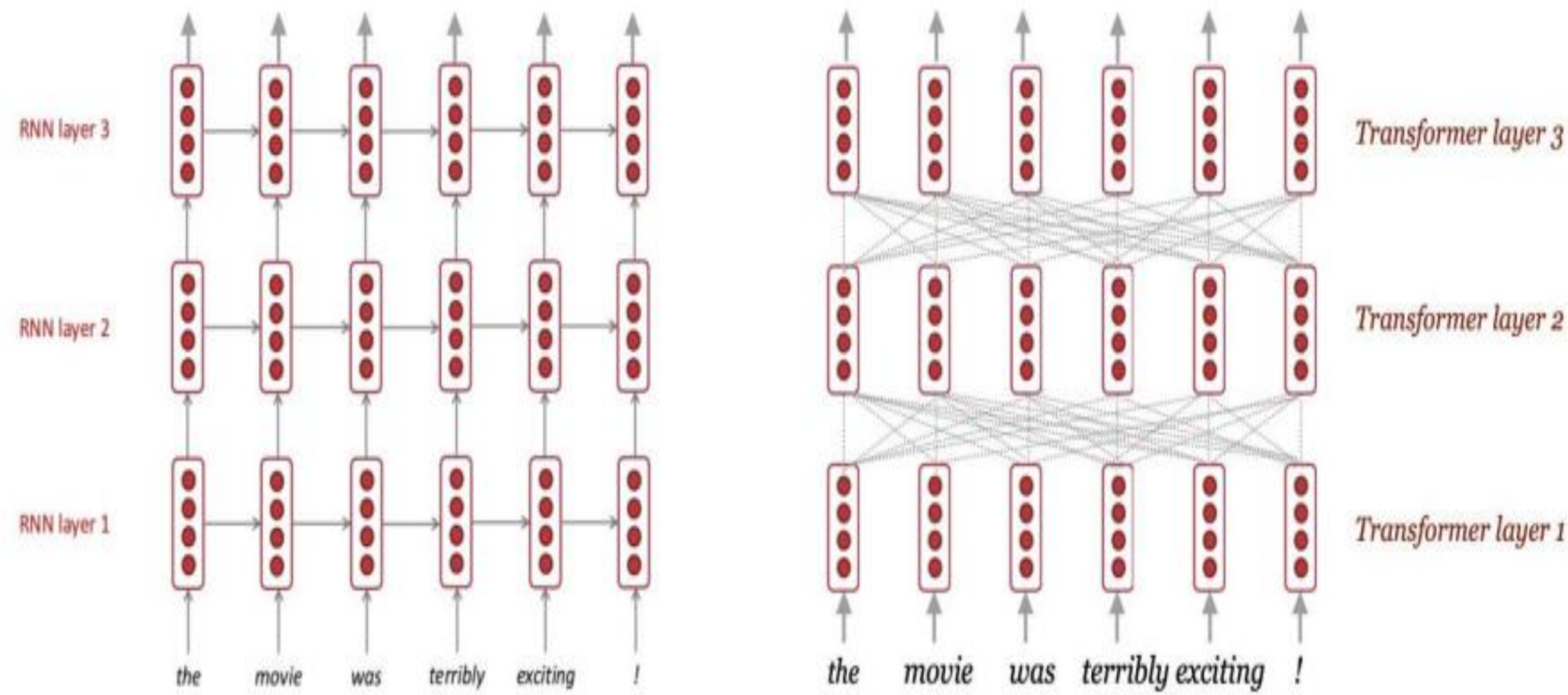


- The attention mechanism enables the Transformer to handle long-range dependencies more effectively than traditional RNNs or CNNs. It allows the model to understand the context of each word in a sentence by considering its relationships with all other words, leading to better performance on various NLP tasks

https://self-supervised.cs.jhu.edu/sp2025/files/slides/10-11.transformers.pdf

RNN vs Transformer

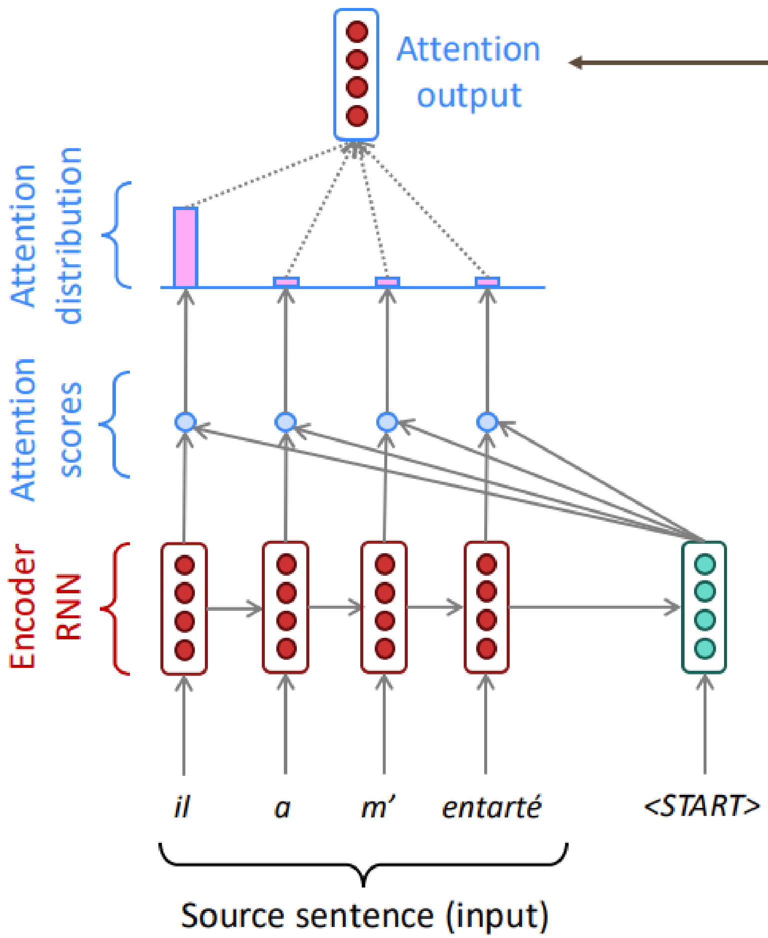
- Notice that self-attention can directly "look" at all tokens and decide which one is useful.



https://self-supervised.cs.jhu.edu/sp2025/files/slides/10-11.transformers.pdf

Transformers for Neural Machine Translation (NMT)

Sequence-to-sequence with attention

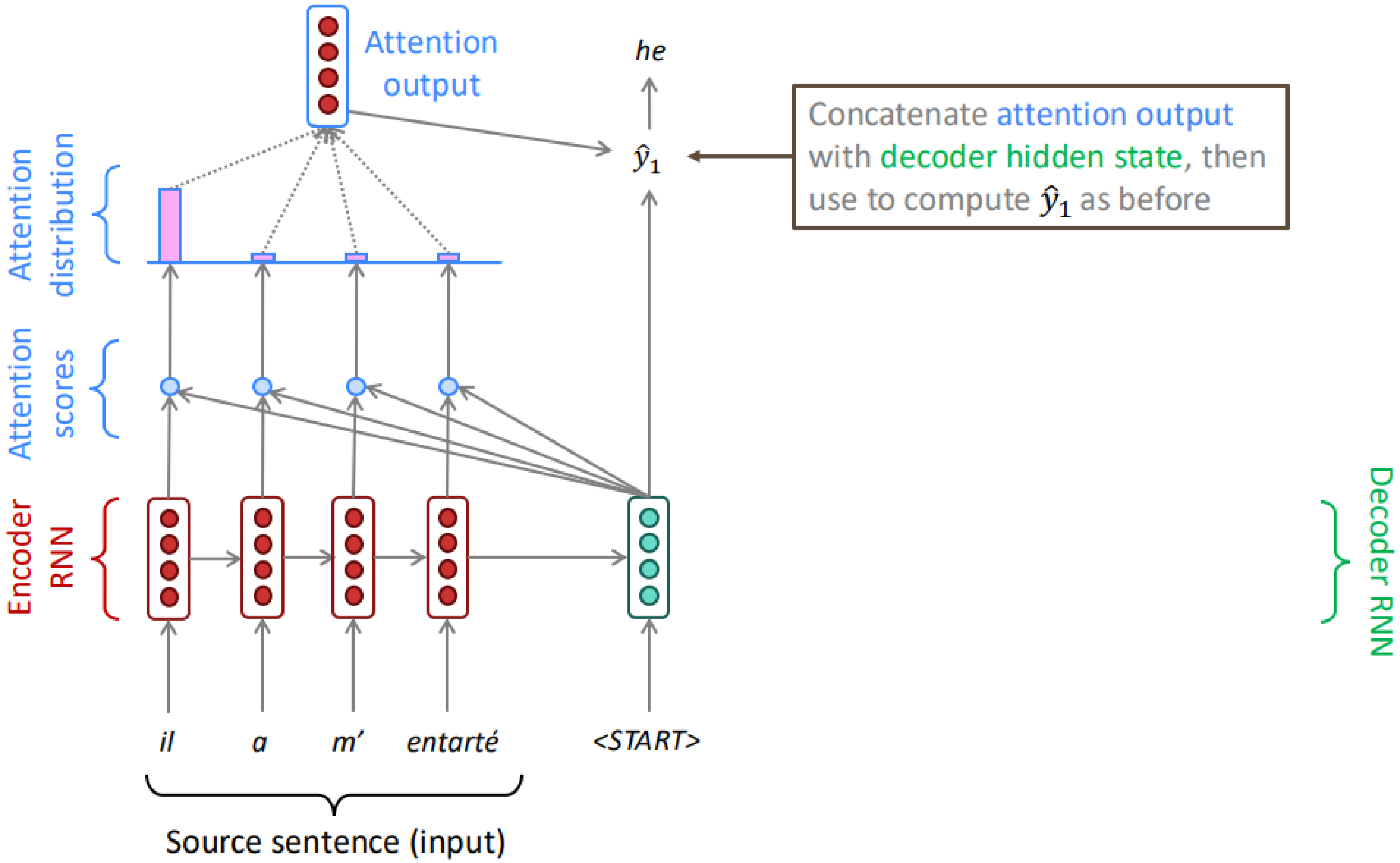


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

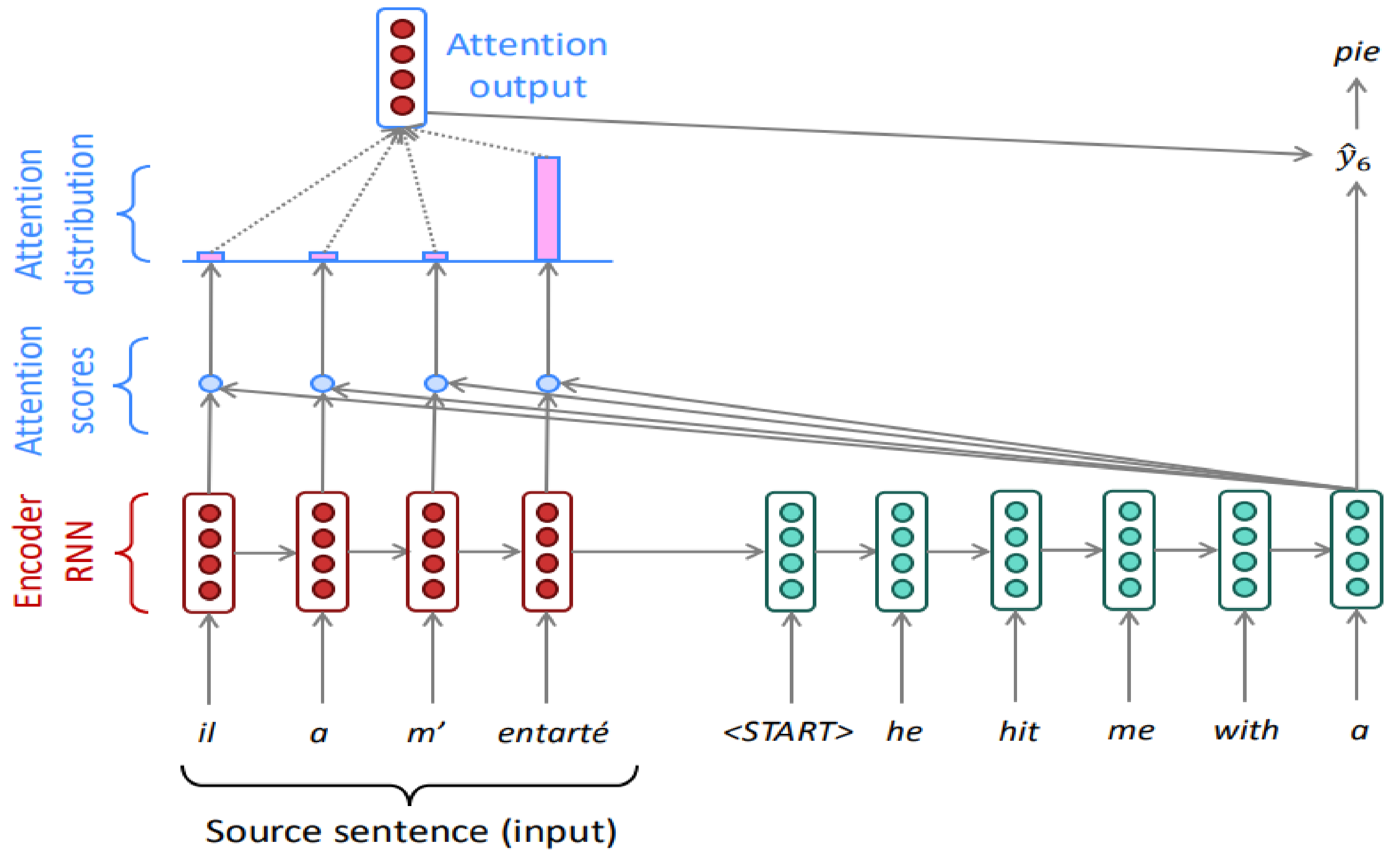
https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture06-fancy-rnn.pdf

Sequence-to-sequence with attention



https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture06-fancy-rnn.pdf

Sequence-to-sequence with attention

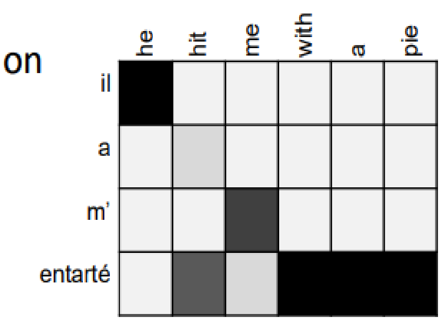


https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture06-fancy-rnn.pdf

Attention is parallelizable, and solves bottleneck issues

Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention provides a more "human-like" model of the MT process
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we see what the decoder was focusing on
  - We get (soft) alignment for free!
  - The network just learned alignment by itself
- (One issue – attention has *quadratic* cost with respect to sequence length)

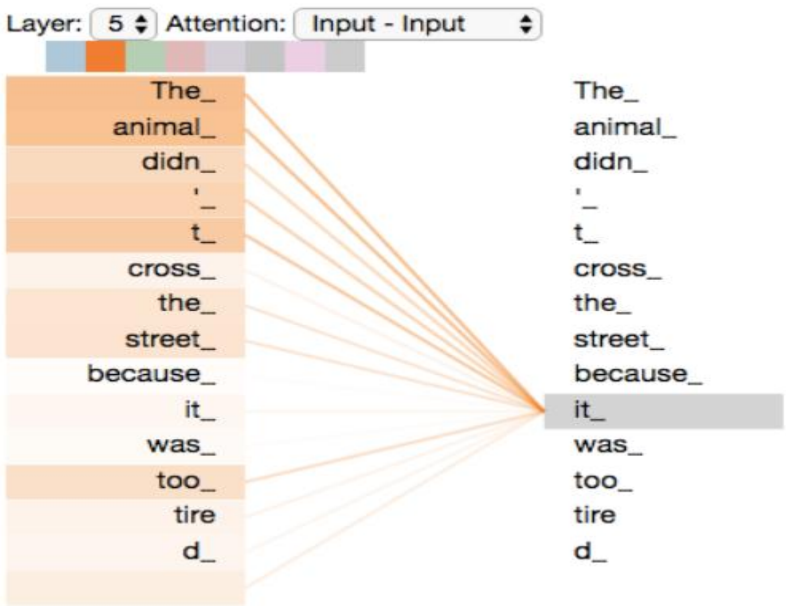


https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture06-fancy-rnn.pdf

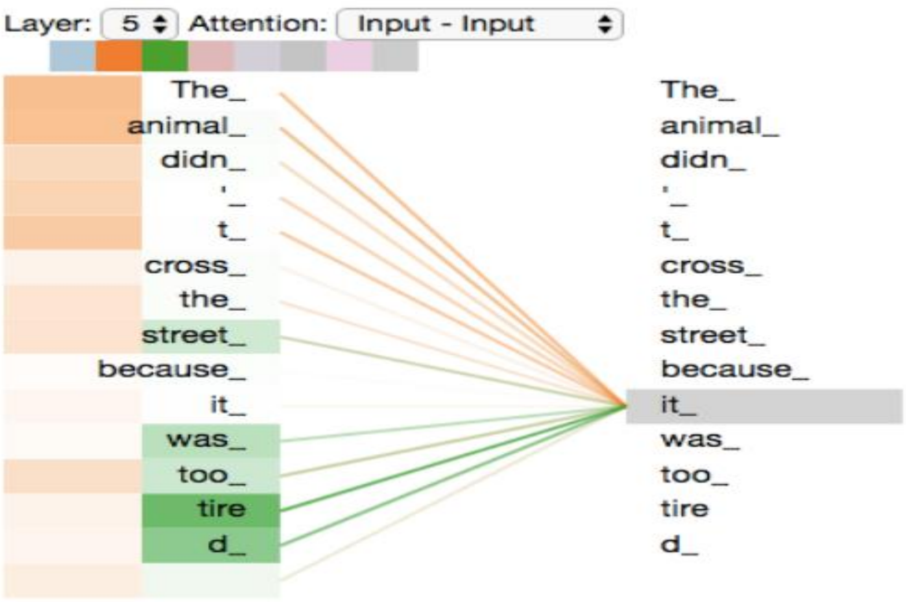
Multihead Attention

- The different words in a sentence can relate to each other in many different ways simultaneously. For example, distinct syntactic, semantic, and discourse relationships can hold between verbs and their arguments in a sentence.
- It would be difficult for a single transformer block to learn to capture all of the different kinds of parallel relations among its inputs.
- Transformers address this issue with multihead self-attention layers.
- These are sets of self-attention layers, called heads, that reside in parallel layers at the same depth in a model, each with its own set of parameters.
- Given these distinct sets of parameters, each head can learn different aspects of the relationships that exist among inputs at the same level of abstraction.

Multihead Attention



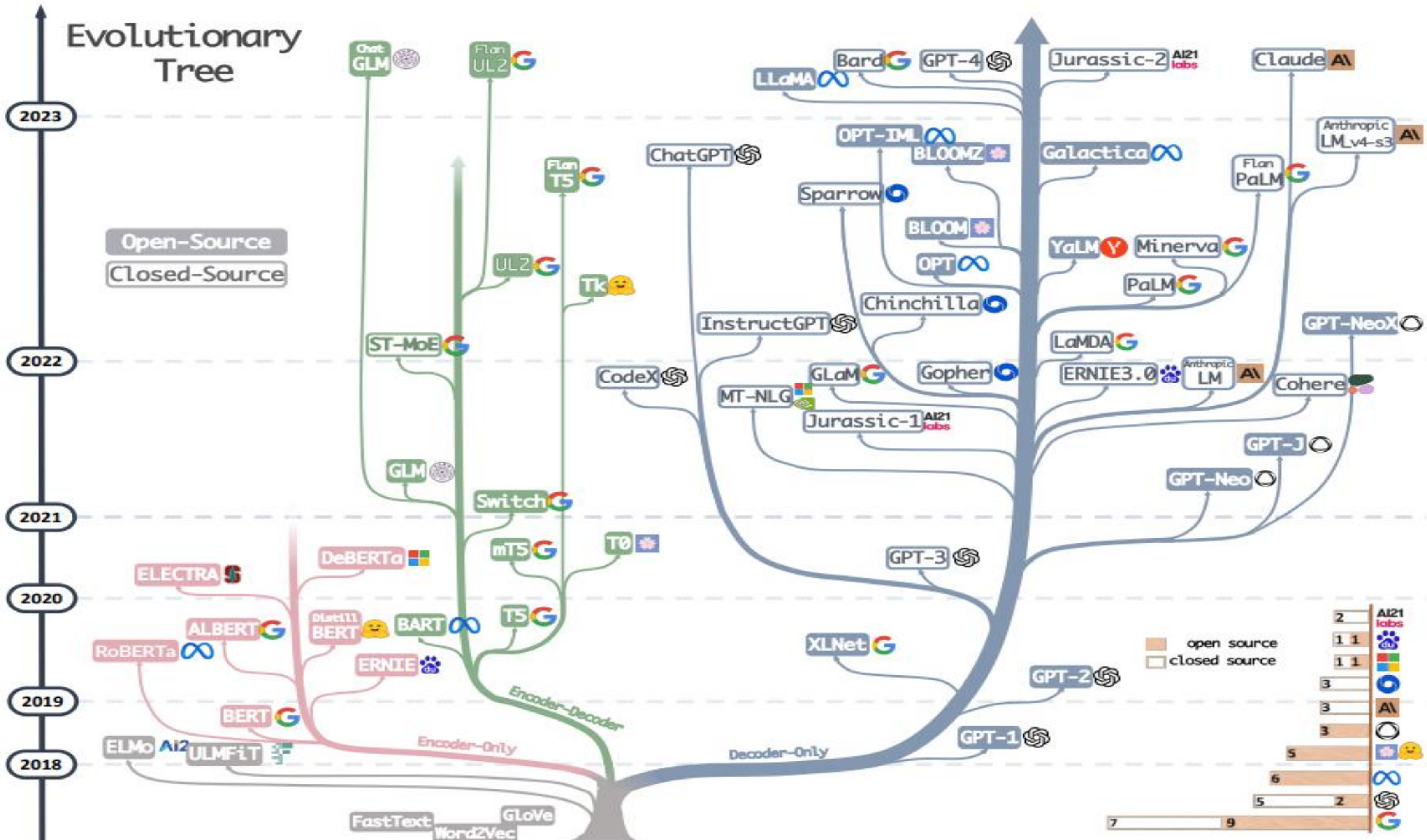
HEAD 1: As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".



HEAD 2: As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".

Yang et al. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, 2023

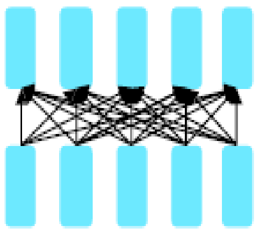
After Transformers



The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models

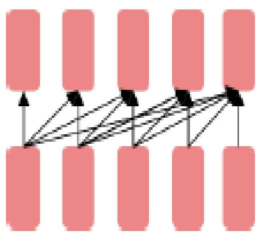
Impact of Transformers

- A building block for a variety of LMs



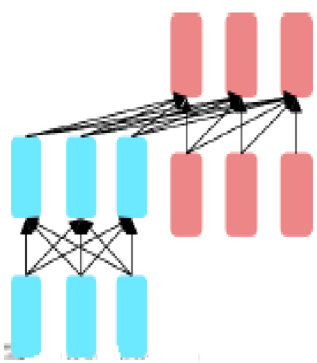
Encoders

- Examples: BERT, RoBERTa, SciBERT.
- Captures bidirectional context.



Decoders

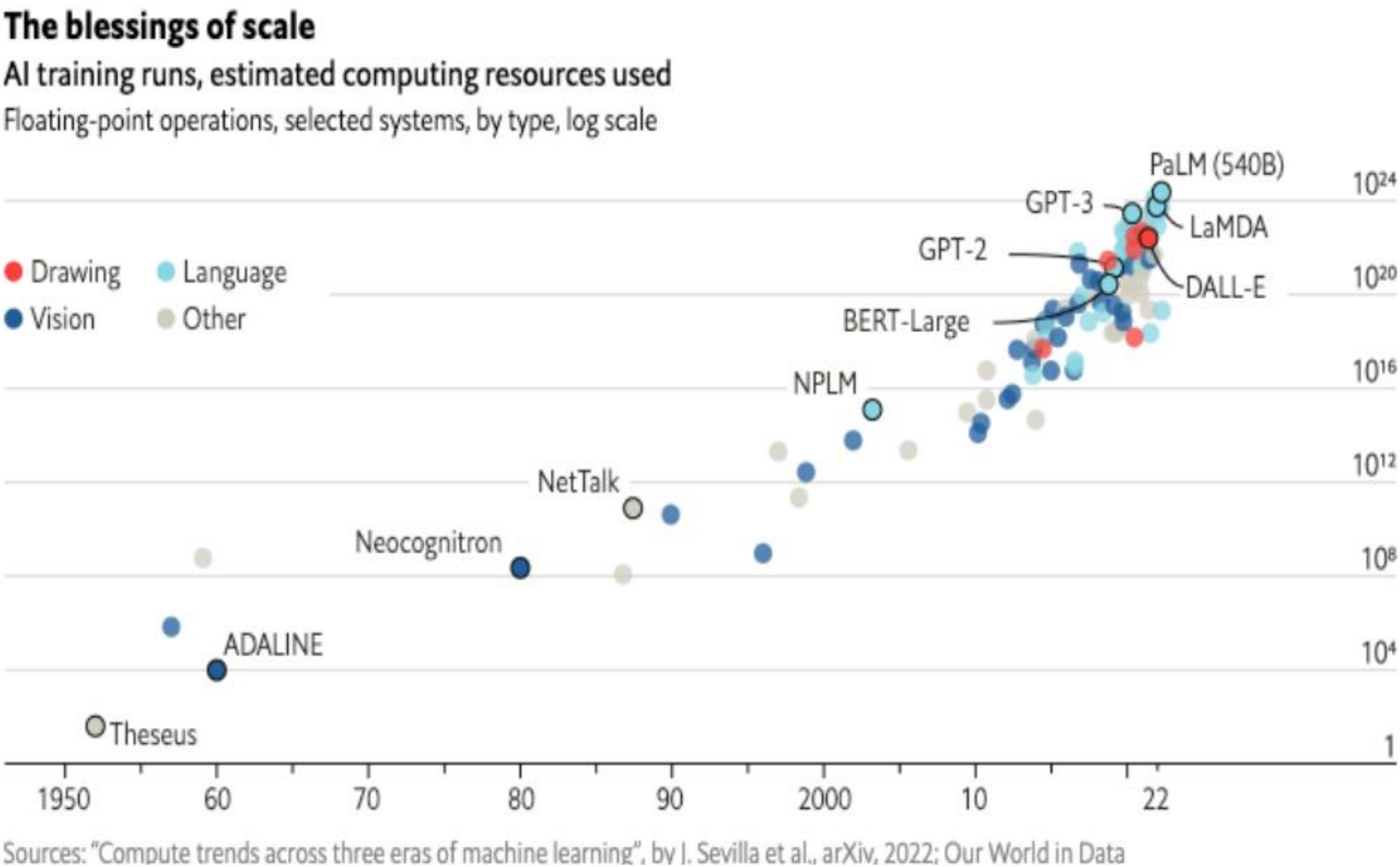
- Examples: GPT-2, GPT-3, LaMDA
- Other name: causal or auto-regressive language model
- Nice to generate from; can't condition on future words



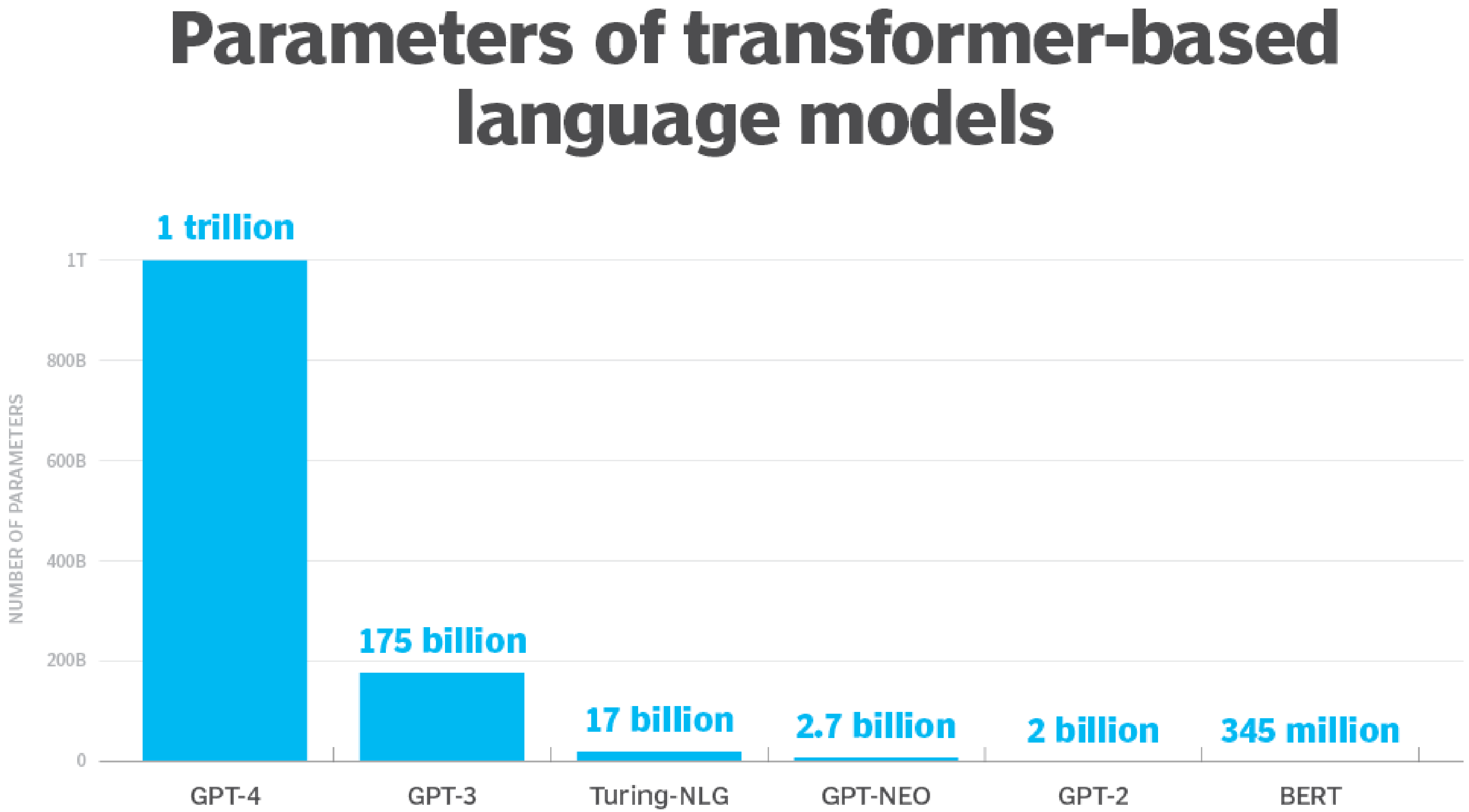
Encoder-Decoders

- Examples: Transformer, T5, Meena
- What's the best way to pretrain them?

Larger and larger models



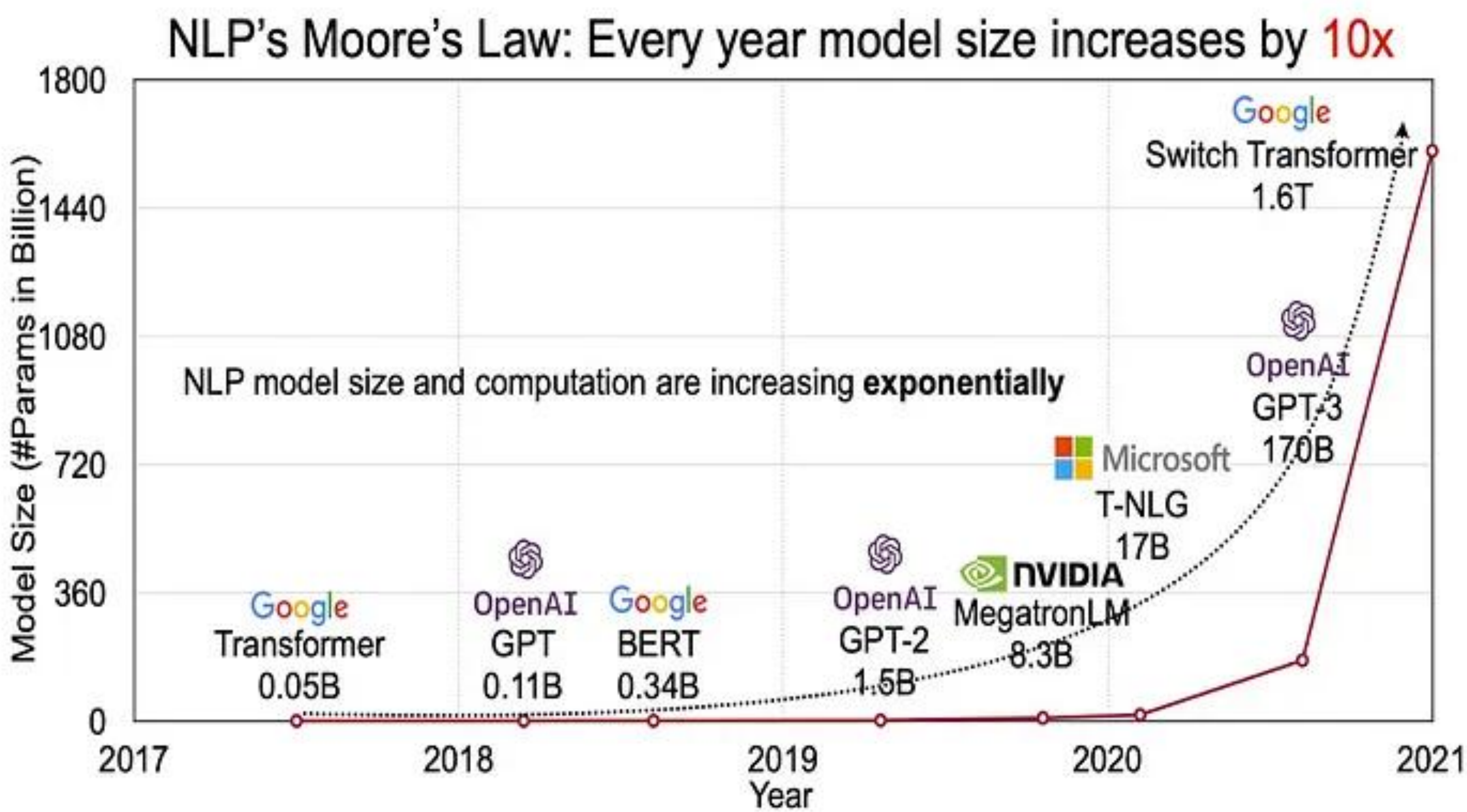
Larger and larger models



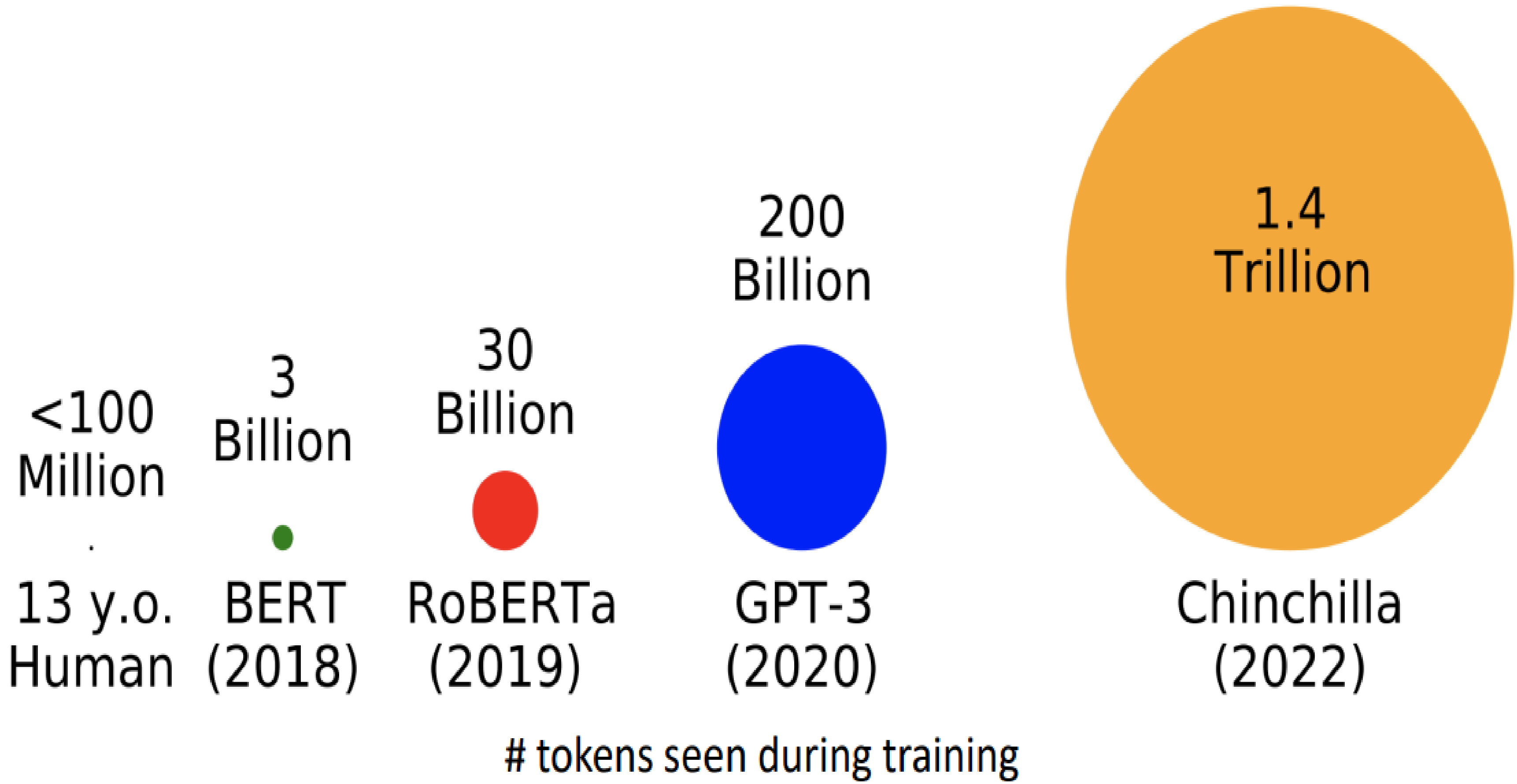
https://www.techtarget.com/whatis/definition/large-language-model-LLM#:~:text=While%20there%20isn't%20a,one%20billion%20or%20more%20parameters

Larger and larger models

Moore's law for the number of transistors on a chip. You can observe the exponential increase in the model size from the below graph. According to Moore's Law, the model size is increasing by a factor of 10 year-on-year.



Trained on more and more data



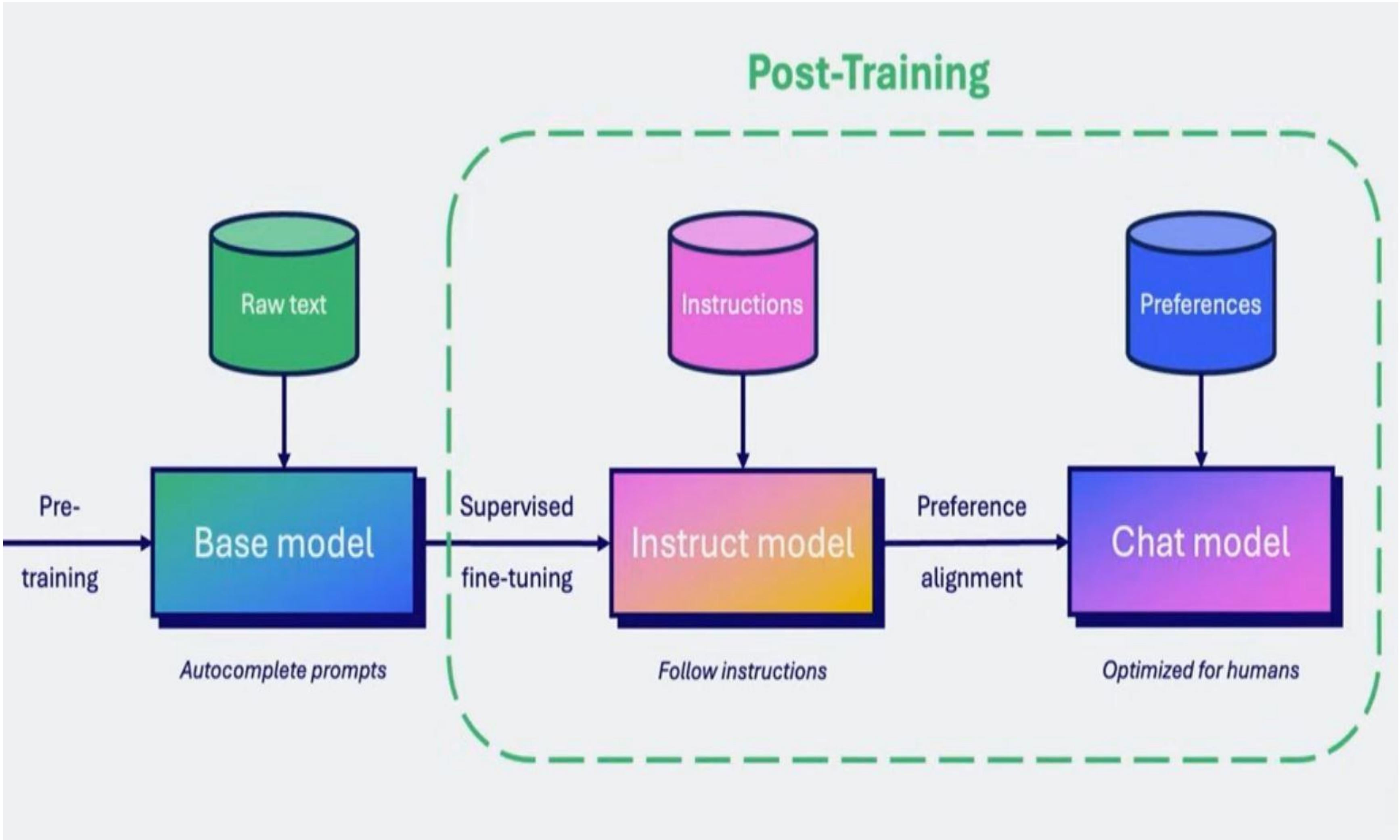
https://babylm.github.io/

Chatbot Arena LLM Leaderboard

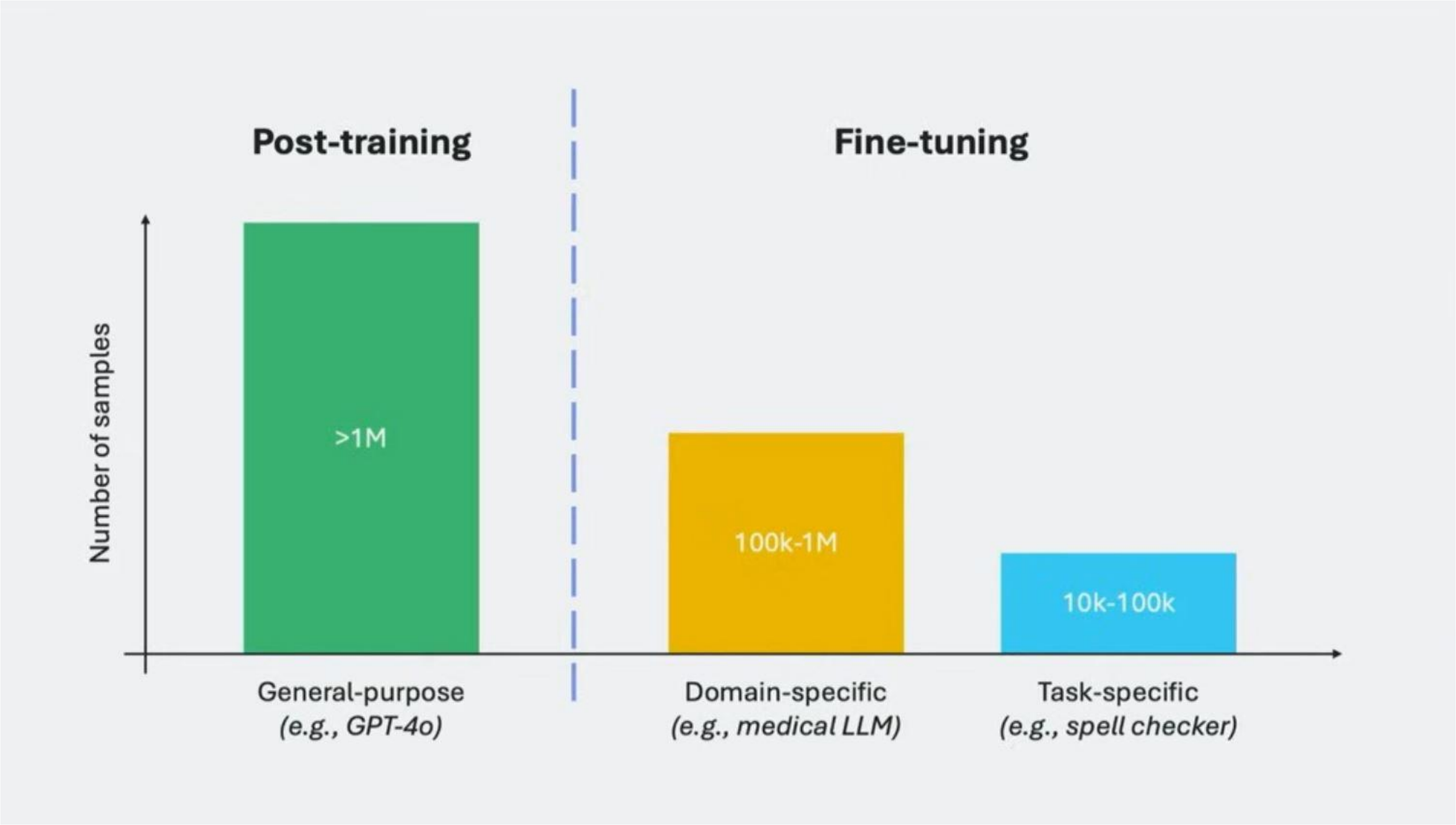
Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	1	GPT-4.5-Preview	1411	+11/-11	3242	OpenAI	Proprietary
1	2	Grok-3-Preview-02-24	1412	+8/-10	3364	xAI	Proprietary
3	2	ChatGPT-4o-latest (2025-01-29)	1377	+5/-4	17221	OpenAI	Proprietary
3	3	Gemini-2.0-Pro-Exp-02-05	1380	+5/-6	15466	Google	Proprietary
3	5	Gemini-2.0-Flash-Thinking-Exp-01-21	1384	+6/-5	17487	Google	Proprietary
6	3	DeepSeek-R1	1363	+8/-6	8580	DeepSeek	MIT
6	10	Gemini-2.0-Flash-001	1357	+6/-5	13257	Google	Proprietary
7	3	o1-2024-12-17	1352	+4/-6	19785	OpenAI	Proprietary
9	7	o1-preview	1335	+4/-3	33167	OpenAI	Proprietary
9	10	Qwen2.5-Max	1336	+7/-5	11930	Alibaba	Proprietary
9	10	o3-mini-high	1329	+8/-6	9102	OpenAI	Proprietary
11	13	DeepSeek-V3	1318	+5/-4	22007	DeepSeek	DeepSeek
12	5	Claude 3.7 Sonnet	1309	+9/-11	4254	Anthropic	Proprietary
12	15	Qwen-Plus-0125	1310	+7/-5	6054	Alibaba	Proprietary
12	16	GLM-4-Plus-0111	1311	+8/-8	6035	Zhipu	Proprietary
13	14	Gemini-2.0-Flash-Lite-Preview-02-05	1308	+5/-5	12774	Google	Proprietary
13	14	o3-mini	1304	+5/-4	15463	OpenAI	Proprietary

https://lmarena.ai/?leaderboard

Training

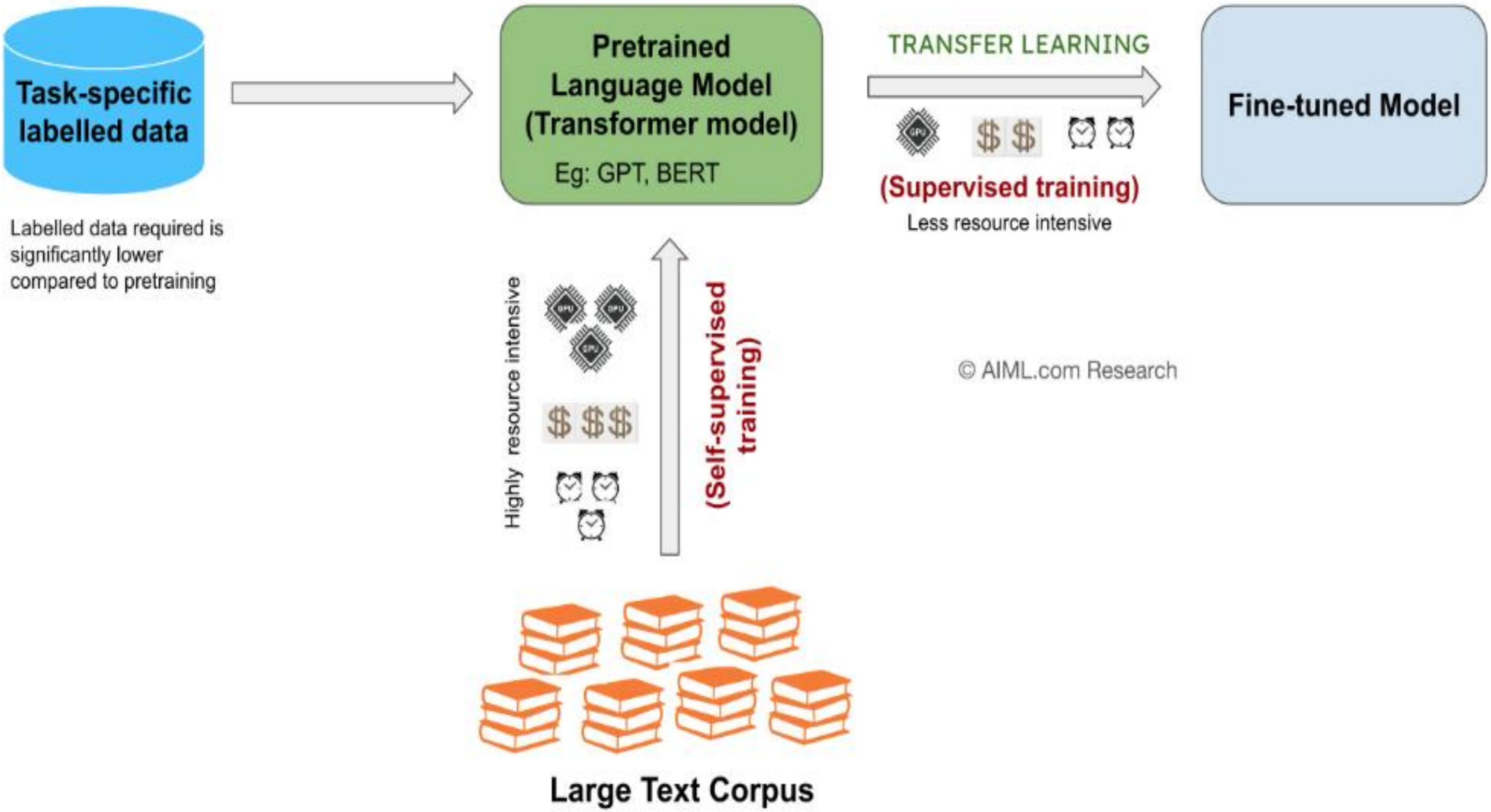

Post-Training
Raw text
Instructions
Preferences
Pre-
training
Base model
Supervised
fine-tuning
Instruct model
Preference
alignment
Chat model
Autocomplete prompts
Follow instructions
Optimized for humans

Training



Post-training
Fine-tuning
Number of samples
>1M
100k-1M
10k-100k
General-purpose
(e.g., GPT-4o)
Domain-specific
(e.g., medical LLM)
Task-specific
(e.g., spell checker)

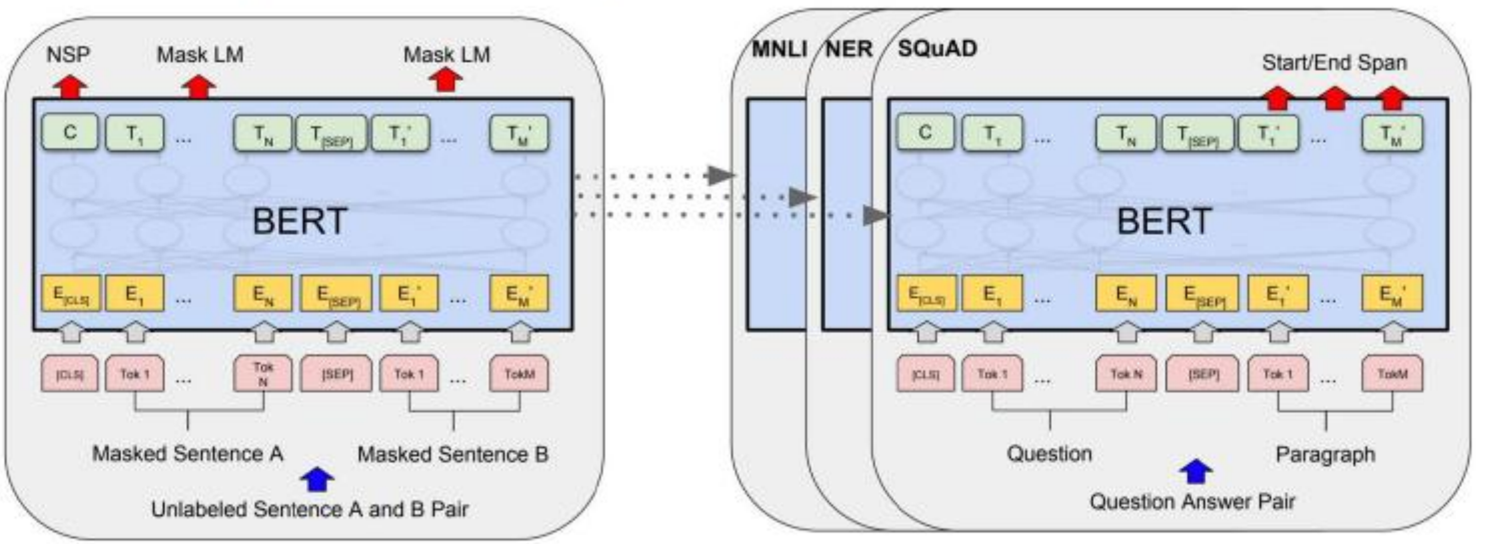
Pretraining and Finetuning



https://aiml.com/what-do-you-mean-by-pretraining-finetuning-and-transfer-learning-in-the-context-of-machine-learning-or-language-modeling/

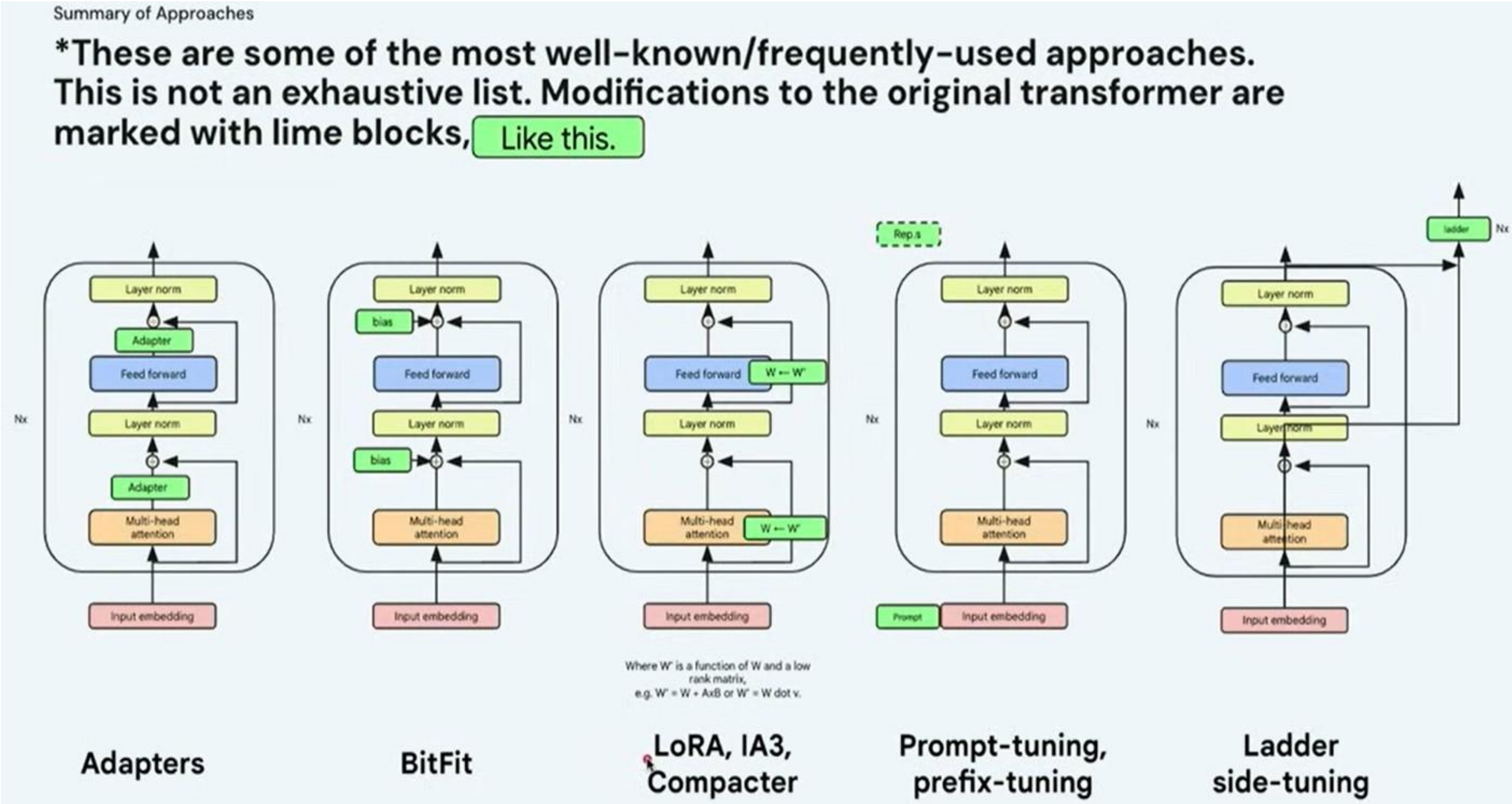
Pretrain once, finetune many times for different tasks

- **Idea:** Make pre-trained model **usable** in **downstream tasks**
- Initialized with pre-trained model parameters
- Fine-tune model parameters using labeled data from downstream tasks



Pre-training Fine-Tuning

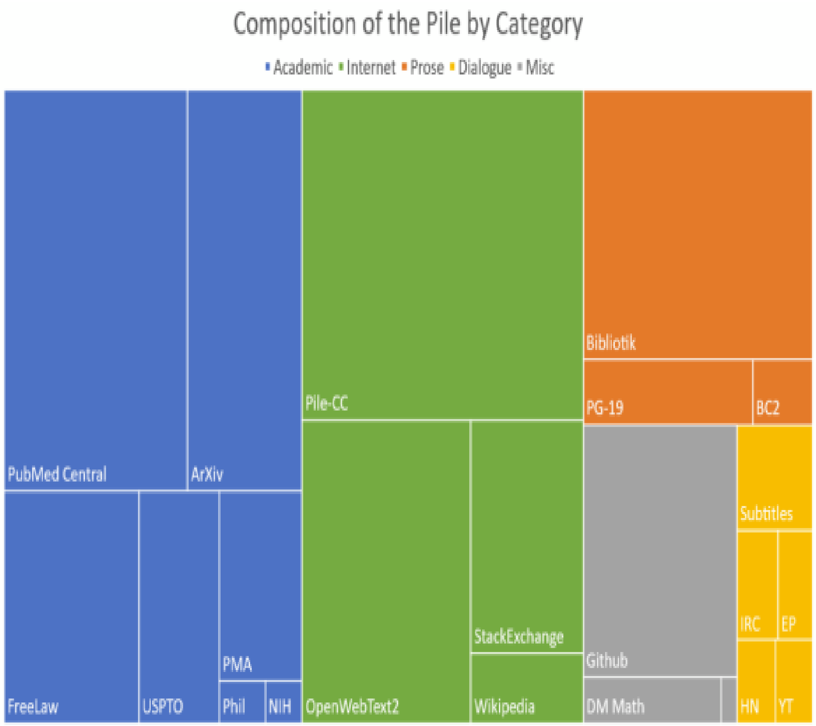
Parameter-efficient fine-tuning (PEFT) methods



The figure summarizes several parameter-efficient fine-tuning (PEFT) methods for adapting large language models (LLMs).Instead of updating all transformer weights (which is expensive), these **methods modify only small parts of the architecture**.The green blocks indicate where each method adds or modifies something in the transformer layer.

Pretraining can be massively diverse

- It's not just about the quantity, but also the incredible *diversity* of internet text data



[Gao+ 20]

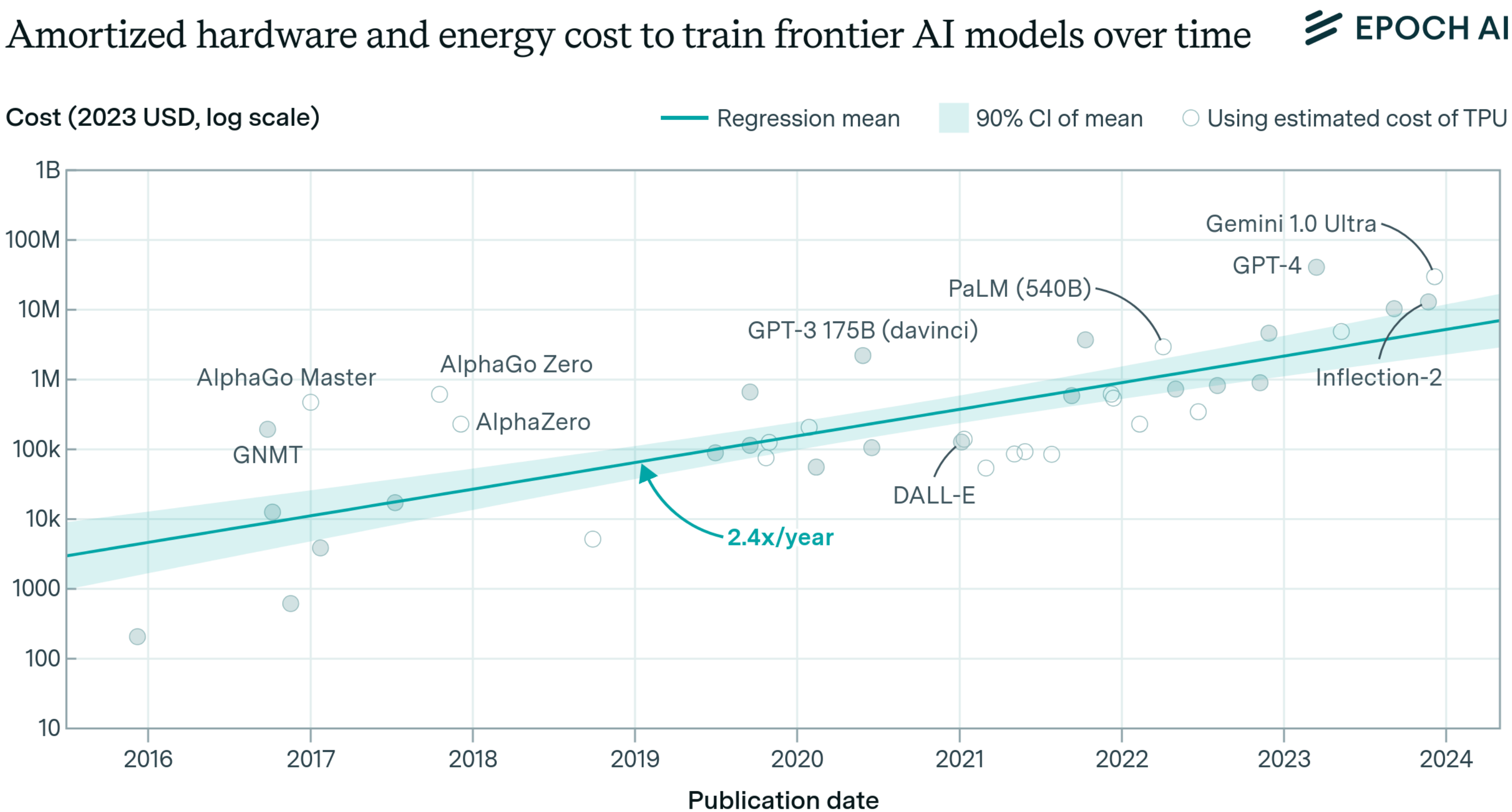
Source	Doc Type	UTF-8 bytes *(GB)*	Documents *(millions)*	Unicode words *(billions)*	Llama tokens *(billions)*
Common Crawl	web pages	9,812	3,734	1,928	2,479
GitHub	code	1,043	210	260	411
Reddit	social media	339	377	72	89
Semantic Scholar	papers	268	38.8	50	70
Project Gutenberg	books	20.4	0.056	4.0	6.0
Wikipedia, Wikibooks	encyclopedic	16.2	6.2	3.7	4.3
Total		**11,519**	**4,367**	**2,318**	**3,059**

[Soldani+ 24]

https://web.stanford.edu/class/cs224n/slides_w25/cs224n-2025-lecture09-pretraining.pdf

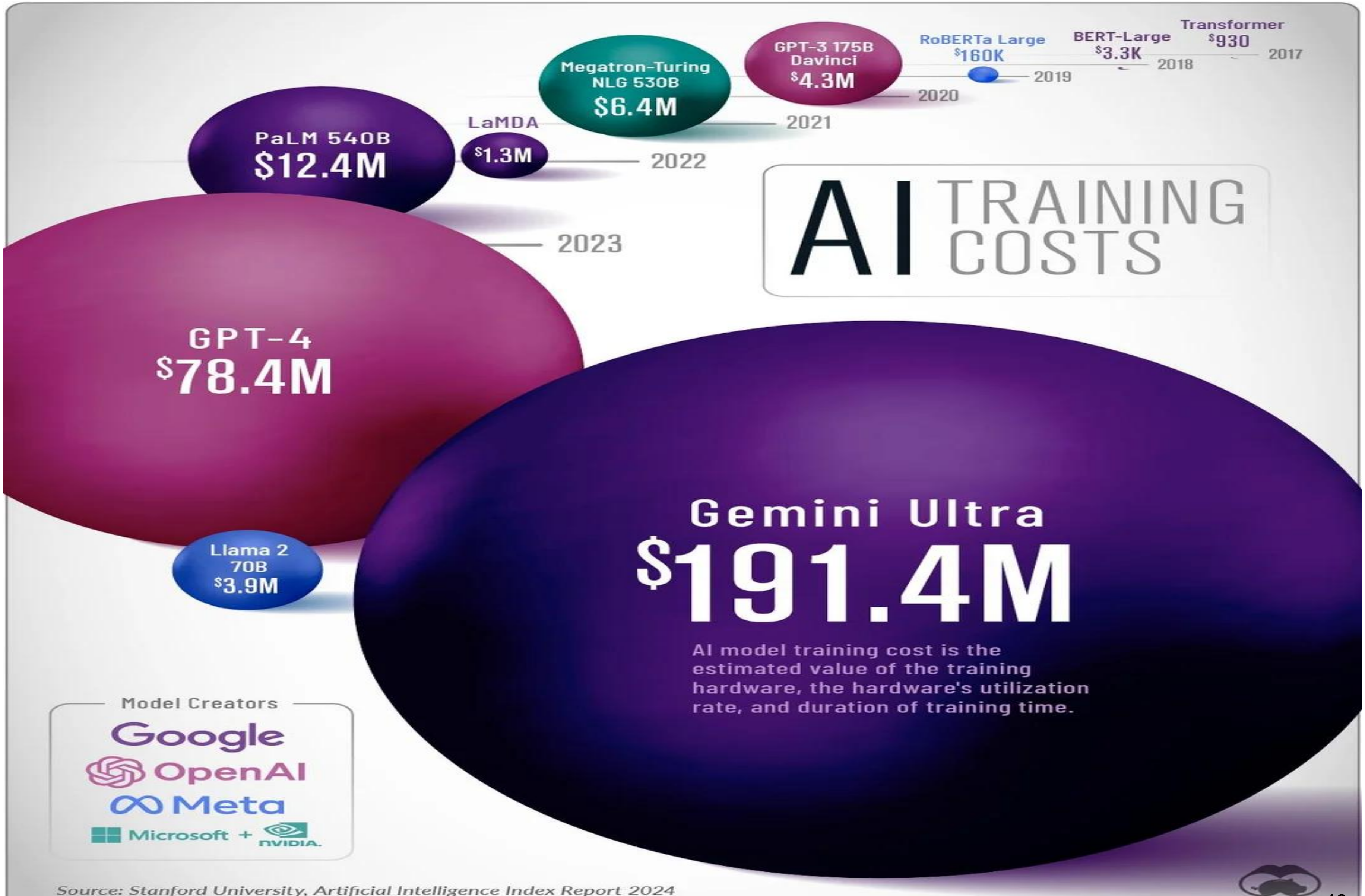
Hardware and Energy Cost

The cost of training frontier AI models has grown by a factor of 2 to 3x per year for the past eight years, suggesting that the largest models will cost over a billion dollars by 2027.



https://epoch.ai/blog/how-much-does-it-cost-to-train-frontier-ai-models

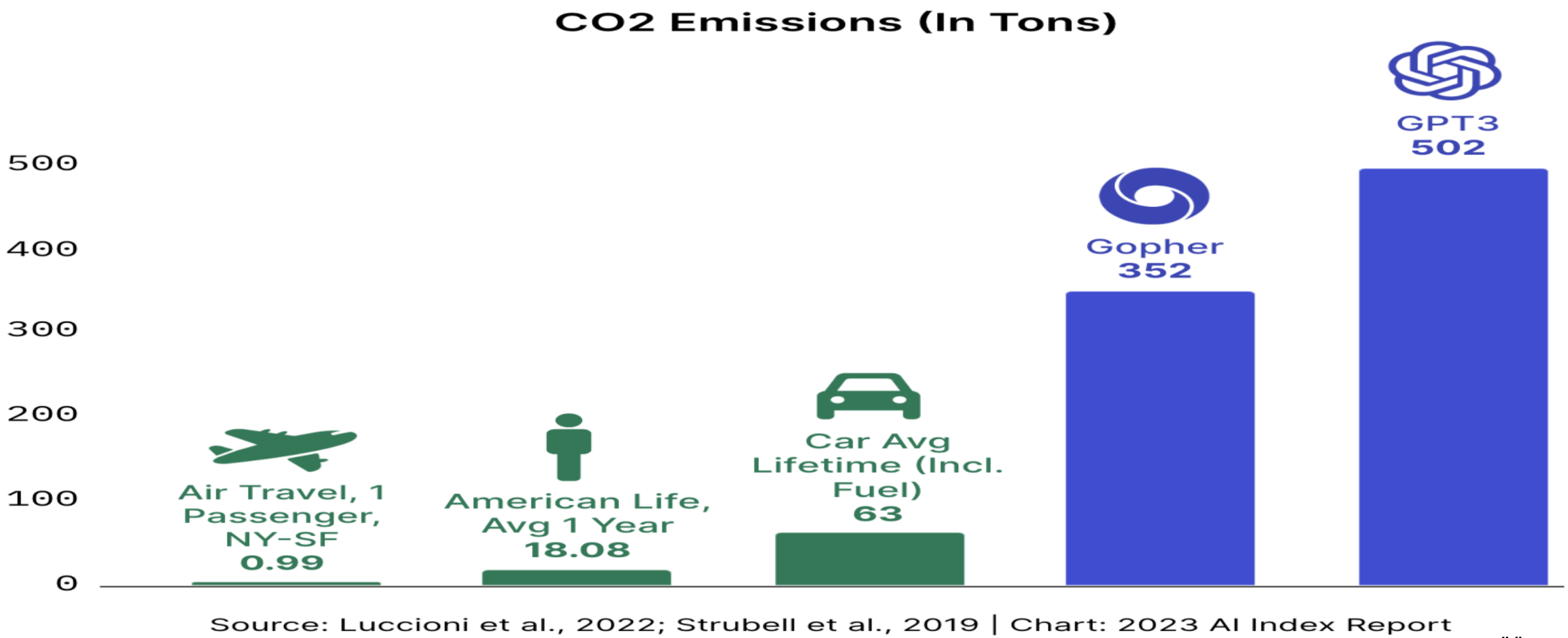
Training Costs



https://www.reddit.com/r/Infographics/comments/1d44tzb/the_training_cost_of_ai_models_over_time/

CO2 Emmisions

- Training a model, especially a large one, requires a large amount of data. This becomes very costly in terms of time and compute resources. It even translates to environmental impact, as can be seen in the following graph.
- Training AI models are **carbon intensive**, and data center construction is **accelerating rapidly**
- Training LLMs like GPT3 produces about 500 metric tons of CO2, equivalent to taking 500 flights from NY to SF.



https://www.austinledzian.com/project/cairn

DeepSeek-V3

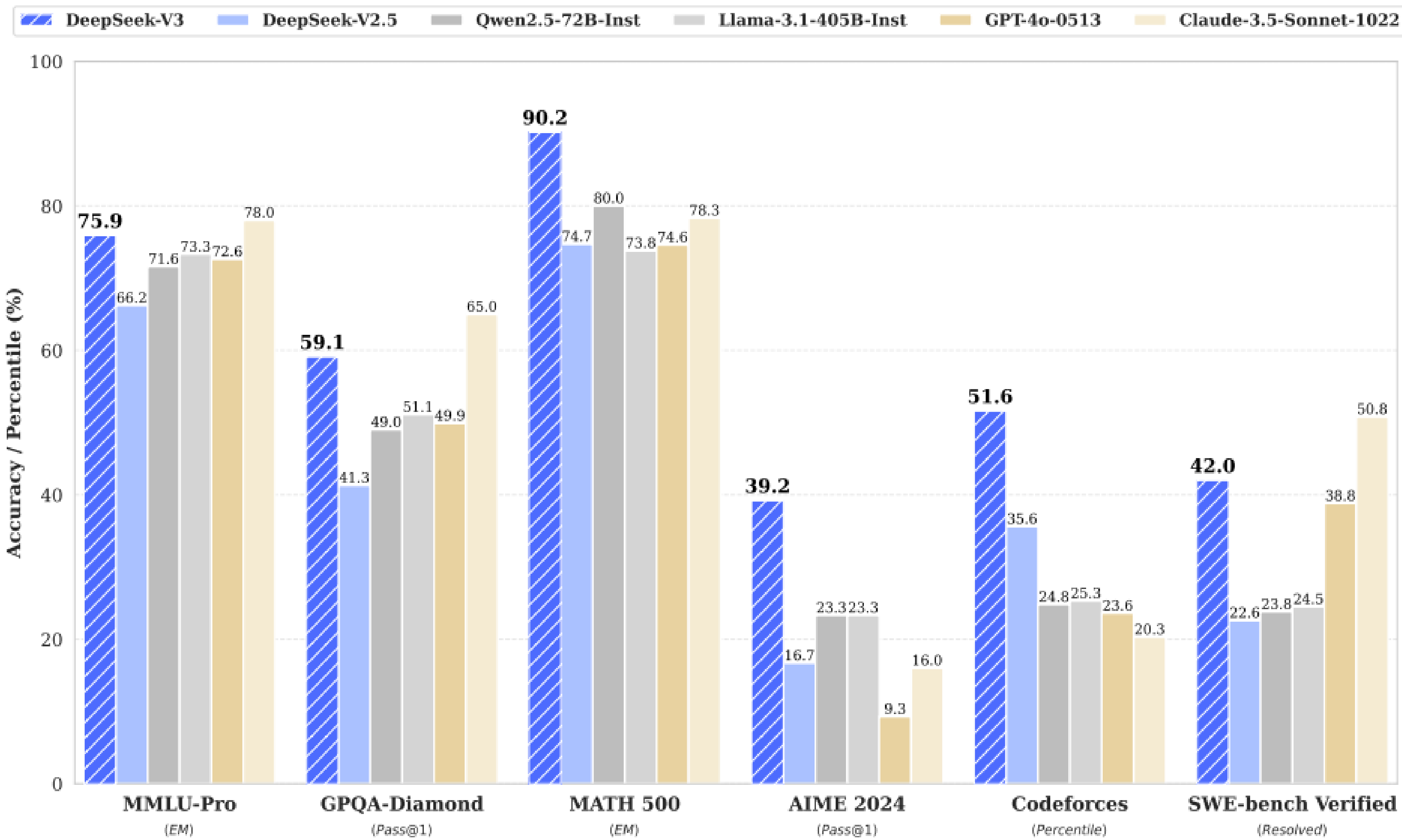


Figure 1 | Benchmark performance of DeepSeek-V3 and its counterparts.

DeepSeek-V3 Technical Report

- we scale up our models and introduce DeepSeek-V3, a large Mixture-of-Experts (MoE) model with 671B parameters, of which 37B are activated for each token.
- During pre-training, we train DeepSeek-V3 on 14.8T high-quality and diverse tokens.
- During the pre-training stage, training DeepSeek-V3 on each trillion tokens requires only 180K H800 GPU hours, i.e., 3.7 days on our cluster with **2048 H800 GPUs**.
- Consequently, our pretraining stage is completed in less than two months **(3.7*14.8 = 54.76 days)**
- Assuming the rental price of the H800 GPU is $2 per GPU hour, our total training costs amount to only $5.576M.

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	$5.328M	$0.238M	$0.01M	$5.576M

Table 1 | Training costs of DeepSeek-V3, assuming the rental price of H800 is $2 per GPU hour.

DeepSeek $6M Cost Of Training Is Misleading

- The $5-6M cost of training is misleading. It comes from the claim that 2048 H800 cards were used for *one* training, which at market prices is upwards of $5-6M.
- Developing such a model, however, requires running this training, or some variation of it, many times, and also many other experiments
- That makes the cost to be many times above that, not to mention data collection and other things, a process which can be very expensive
- Also, 2048 H800 cost between $50-100M.

https://therecursive.com/martin-vechev-of-insait-deepseek-6m-cost-of-training-is-misleading/

AI Arms Race

- Demis Hassabis, Google's artificial intelligence (AI) chief:
  - In an interview with Bloomberg Television, Hassabis, who leads Google's DeepMind, called the idea that the Chinese startup spent so little to develop an AI system that rivals American tech giants "exaggerated and a little bit misleading."
  - His comments come at a time when, as PYMNTS wrote last week, the "AI arms race is getting pricey," with Google, Meta Microsoft and Amazon planning to collectively spend at least $320 billion on capital expenditures in 2025, the bulk of it for AI.
  - Meta's budget for capital expenditures could reach as high as $65 billion,
  - while Google has set aside $75 billion, primarily for data centers, servers and networking infrastructure.
  - Amazon projects it will spend $100 billion,
  - while Microsoft is booking $80 billion to construct data centers, train AI models and launch AI and cloud-based applications.

https://www.pymnts.com/artificial-intelligence-2/2025/google-ai-head-reports-of-deepseeks-low-costs-are-misleading/

Teşekkürler