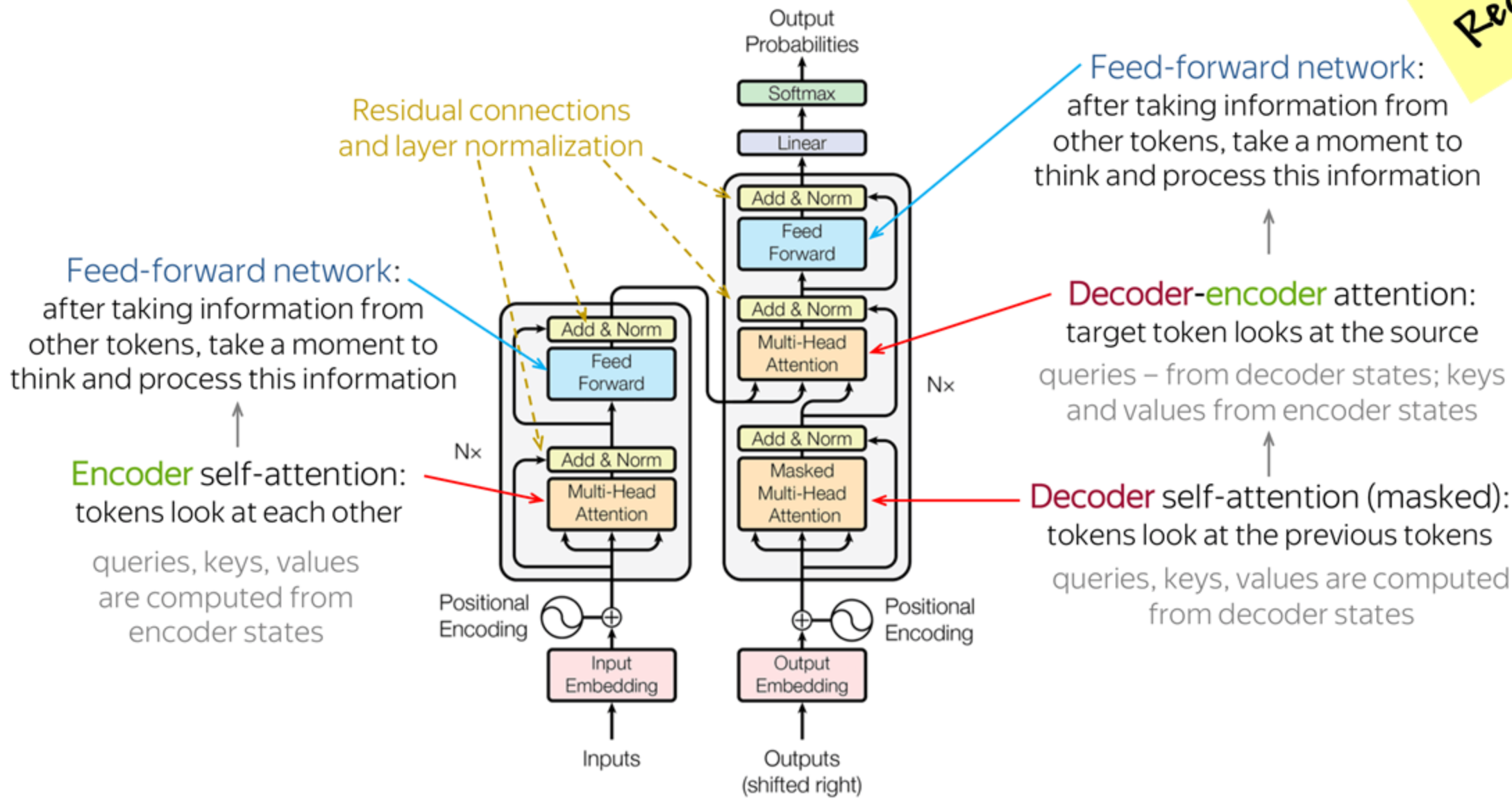


CSCI 5541: Natural Language Processing

Lecture 11: Pretraining Paradigm and Scaling Law



Some slides borrowed from Anna Goldie (Google Brain)





Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hrs)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x V100 (1 day)
GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100
GPT-3	96	12288	96	175B	694GB	?

Brown et al, "Language Models are Few-Shot Learners", arXiv 2020

<http://hal.cse.msu.edu/teaching/2020-fall-deep-learning/14-nlp-and-transformers/#/22/0/9>



Agenda

- ❑ What can we learn from reconstructing the input in the pretrained models?
- ❑ Pretraining for three types of architectures
 - Encoder-only
 - Decoder-only
 - Encoder-Decoder
- ❑ GPT3, in-context learning, and VERY large language models
- ❑ Law of scale
- ❑ The end of pretrain-scaling and the limits of next token prediction



Input Reconstruction



What can we learn from reconstructing the input?

BERT-Base	12	768	12	110M	13GB
BERT-Large	24	1024	16	340M	13GB

University of Minnesota is located in _____, Minnesota.






minneapolis	0.950
bloomington	0.024
duluth	0.017
austin	0.003
rochester	0.002

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

University of Minnesota is located in _____, California.






 minneapolis	0.584
 sacramento	0.116
 bloomington	0.103
 berkeley	0.034
 davis	0.027

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

I put ___ fork down on the table.

	0.982
my	
	0.017
the	
	0.000
her	
	0.000
his	
	0.000
a	

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

The woman walked across the street, checking for traffic
over ___ shoulder

her	0.992
one	0.003
his	0.002
the	0.001
my	0.001

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

I went to the ocean to see the fish, turtles, seals, and _____.

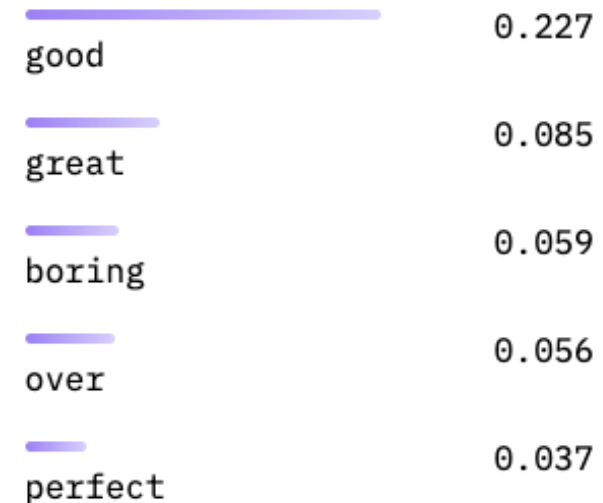
dolphins	0.375
whales	0.324
birds	0.042
sharks	0.038
penguins	0.038

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

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



<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____

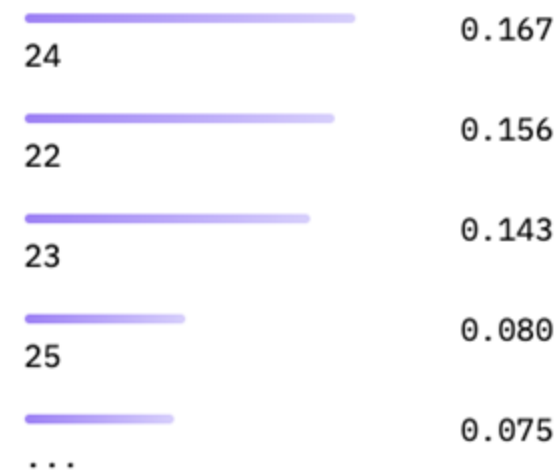
room	0.626
house	0.121
kitchen	0.090
apartment	0.017
table	0.016

<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _ _ _ _



<https://huggingface.co/bert-large-uncased>



What can we learn from reconstructing the input?

What I laern from today's NLP class is how taaasty _ _ _ _ is

it	0.332
she	0.263
he	0.124
that	0.035
this	0.017

<https://huggingface.co/bert-large-uncased>



Pretraining



Pretrained word (type) embeddings

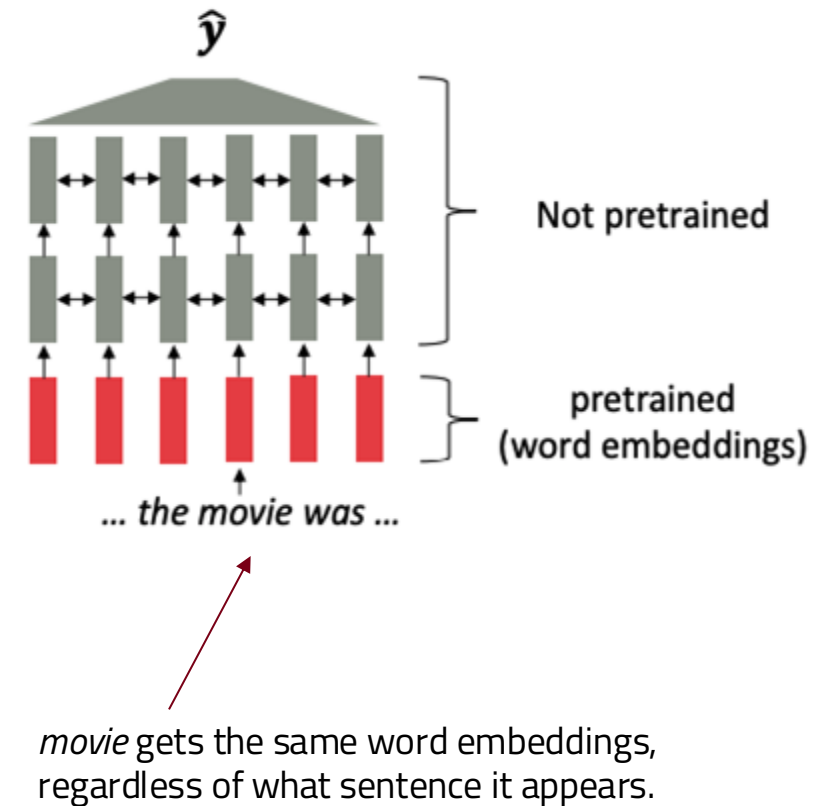


□ Before 2017:

- Start with pretrained **word embeddings (no context!)**
- Learn how to incorporate context in an LSTM or RNN while training on the task.

□ Some issues to think about:

- The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network are **randomly** initialized!



Pretrained **whole (token)** embeddings

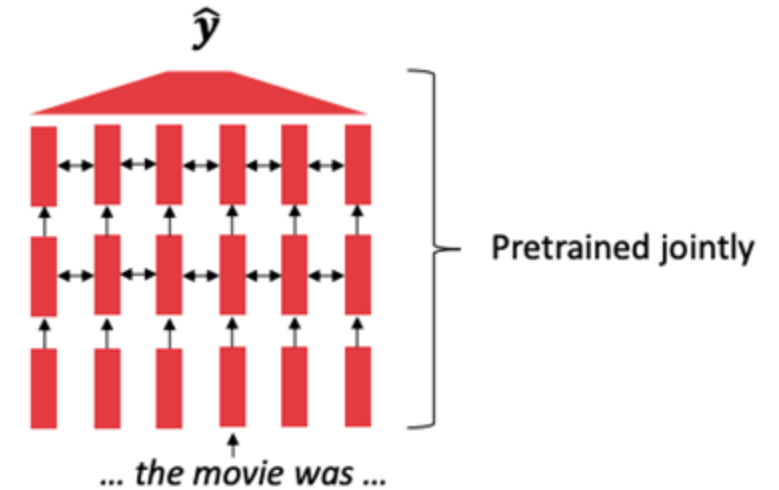


□ In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via pretraining.
- Pretraining methods **hide parts of the input** from the model, then train the model to **reconstruct** those parts

□ This has been exceptionally effective at building strong:

- representations of language
- parameter initializations for strong NLP models.
- probability distributions over language that we can sample from



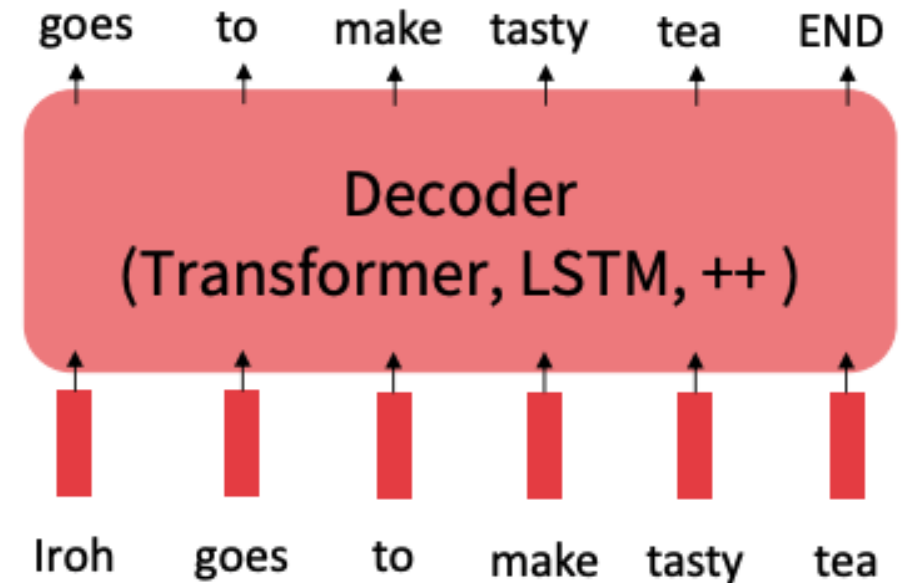
This model has learned how to represent entire sentences through pretraining



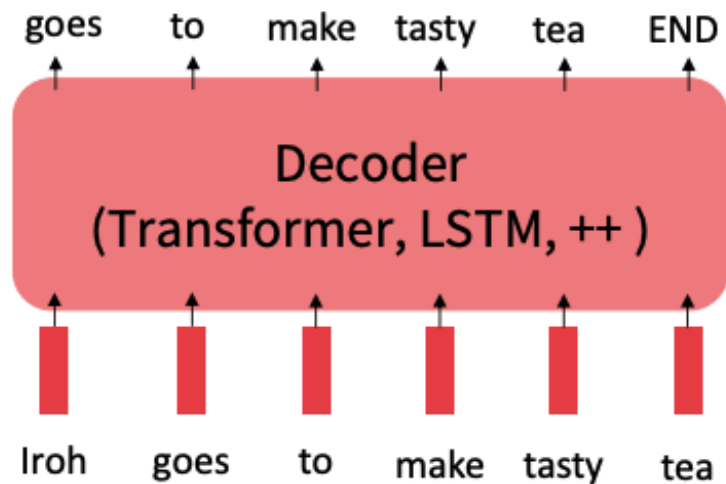
Pretraining through language modeling



- Recall the language modeling task:
 - Model the probability distribution over words given their past contexts.
- Pretraining through language modeling:
 - Train a neural network to perform language modeling on a large amount of text.
 - Save the network parameters.

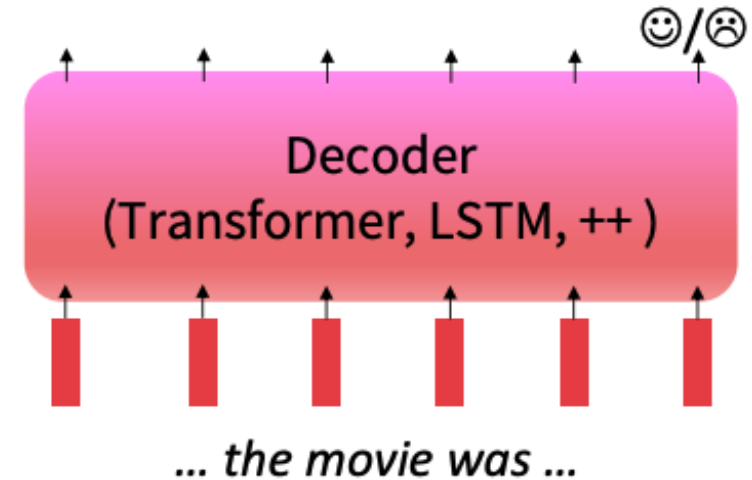


The Pretraining / Finetuning Paradigm



Step 1: Pretrain (on language modeling)

Lots of text; learn general things!
Serve as parameter initialization.

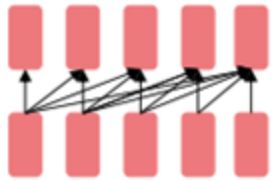


Step 2: Finetune (on your task)

Not many labels; adapt to the task!

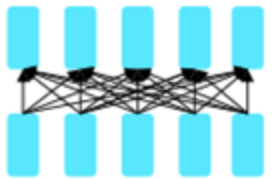


Pretraining for three types of architectures



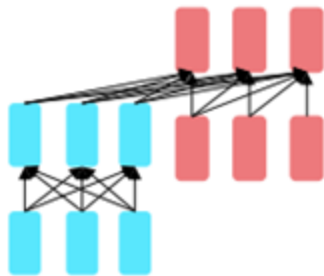
Decoders

- ❑ Simple left-to-right language models!
- ❑ Nice to generate from; can't condition on future words
- ❑ Examples: GPT-2, GPT-3, LaMDA



Encoders

- ❑ Gets bidirectional context – can condition on future!
- ❑ Masked language models
- ❑ Examples: BERT, RoBERTa

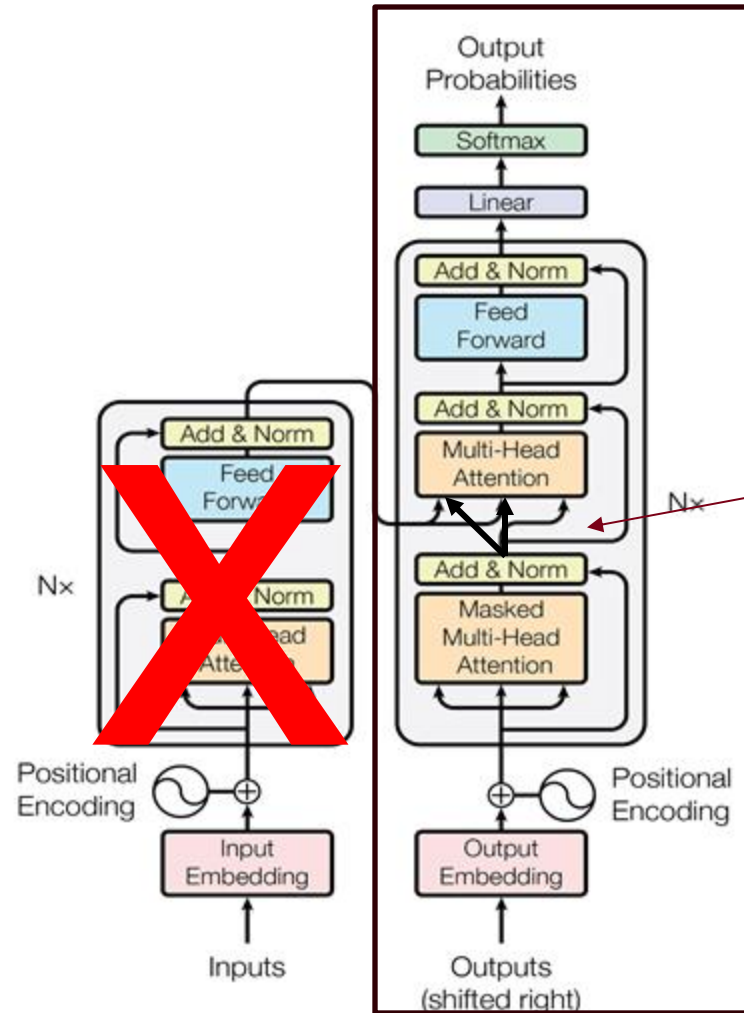


**Encoder-
Decoders**

- ❑ Good parts of decoders and encoders?
- ❑ What's the best way to pretrain them?
- ❑ Examples: T5, BART



Decoder - Only



Multi-Head Attention now utilizes KVs from the decoder model

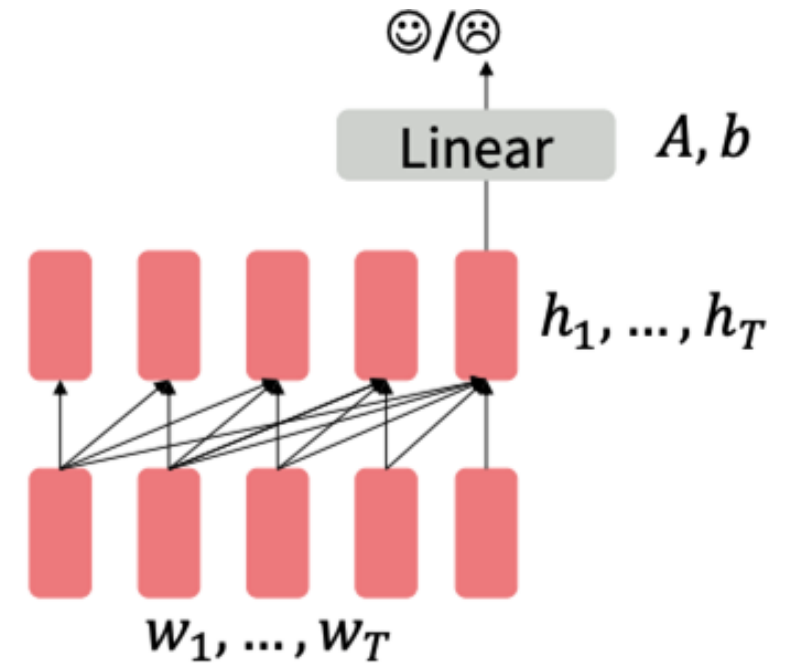


Pretraining and finetuning decoders

- When using language model pretrained decoders, we can ignore that they were trained to model
- We can finetune them by training a **classifier** on the last word's hidden state.
- (Words $\{w_1, w_2, \dots, w_T\}$ as inputs)

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$y \sim Ah_T + b$$

- Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

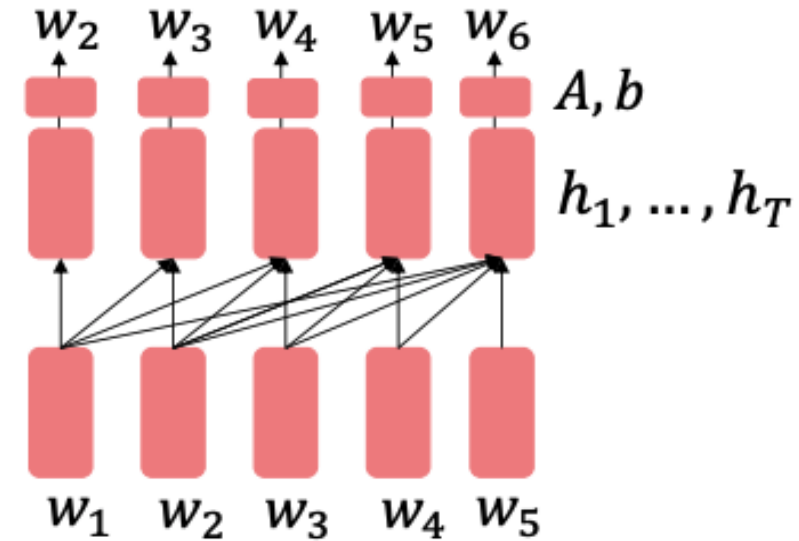
Pretraining and finetuning decoders

- It's natural to pretrain decoders as **language models** and then use them as **generators**, finetuning the decoder: $P_{\theta}(w_t | w_{1:t-1})$

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

where A, b were pretrained in the language model!

- This is helpful in tasks where the output is a sequence with a vocabulary like that at pretraining time!
 - Dialogue (context = dialogue history)
 - Summarization (context=document)



[Note how the linear layer has been pretrained.]



Generative Pretrained Transformer (GPT) (Radford et al., 2018)

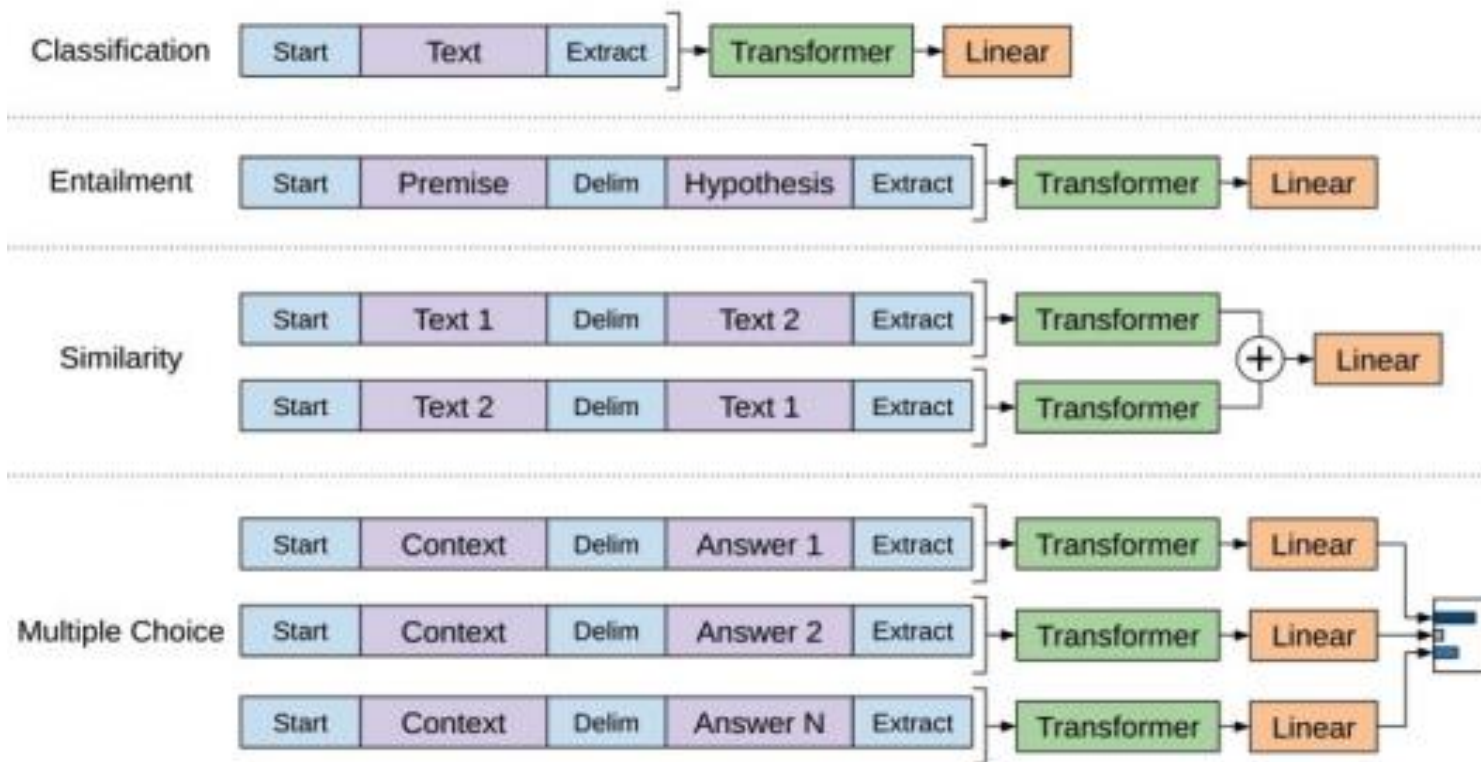
- ❑ 2018's GPT was a big success in pretraining a decoder!
 - Transformer decoder with 12 layers
 - 768-dimensional hidden states
 - 3072-dimensional feed-forward hidden layers
 - Byte-pair encoding with 40,000 merges
 - Trained on BookCorpus: over 7000 unique books.
 - ✓ Contains long spans of contiguous text, for learning long-distance dependencies.

Transformer



Generative Pretrained Transformer (GPT) (Radford et al., 2018)

□ How do we format inputs to our decoder for finetuning tasks?



The linear classifier is applied to the representation of the [EXTRACT] token.



Generative Pretrained Transformer (GPT) (Radford et al., 2018)

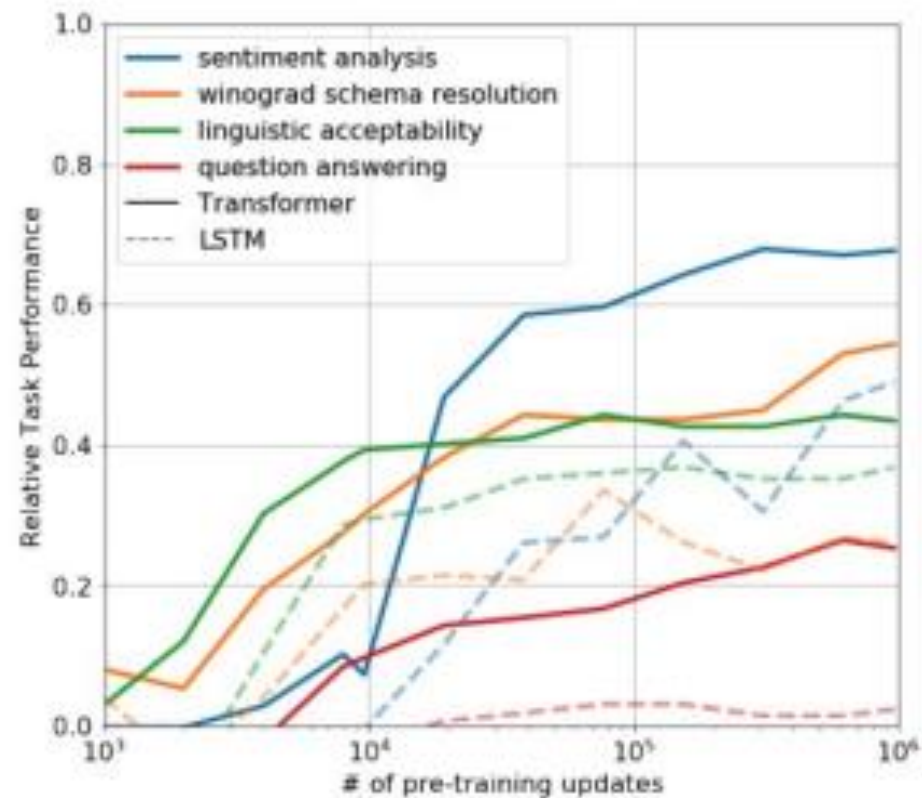
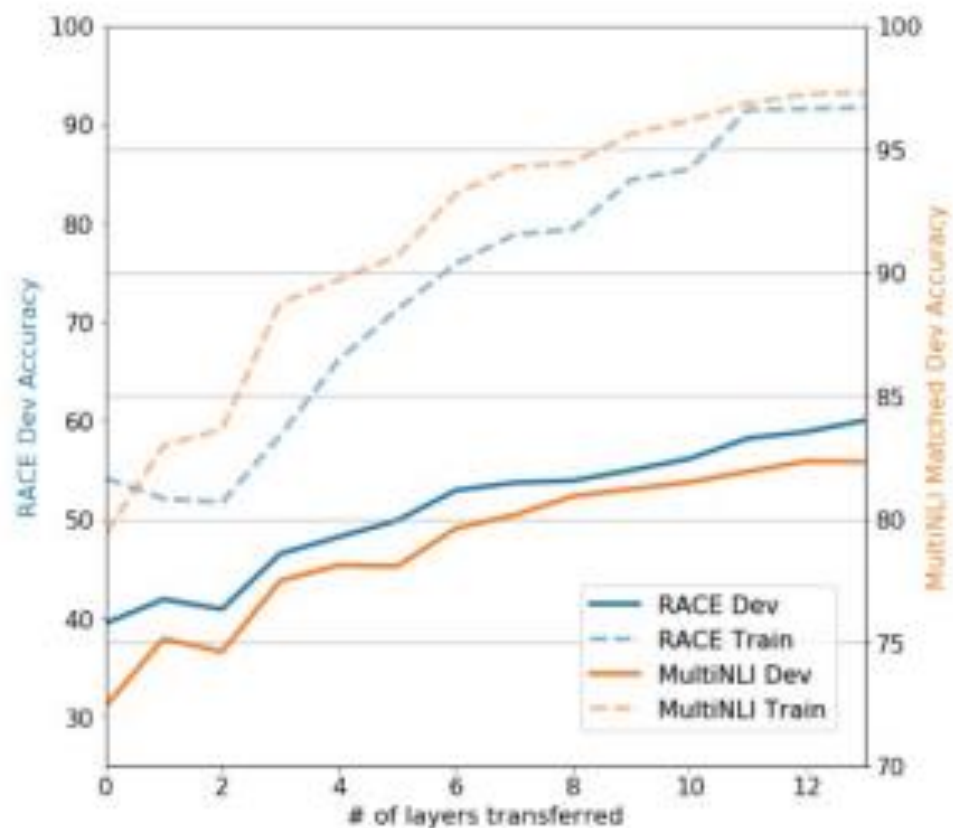
- ❑ GPT results on various natural language inference datasets.
- ❑ Simple but easily adaptable paradigm wins

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	61.7
Finetuned Transformer LM (ours)	82.1	81.4	89.9	88.3	88.1	56.0



Effect of Pretraining in GPT

□ More layers or data always help



Increasingly convincing generations (GPT2) (Radford et al., 2018)

- **GPT-2**, a larger version of GPT trained on more data, was shown to produce relatively convincing samples of natural language

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.



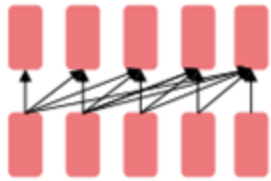
Generative Pretrained Transformer (GPT)

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hrs)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x V100 (1 day)
GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100
GPT-3	96	12288	96	175B	694GB	?

Brown et al, "Language Models are Few-Shot Learners", arXiv 2020

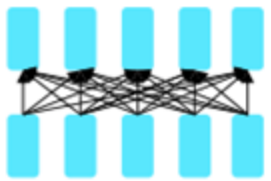


Pretraining for three types of architectures



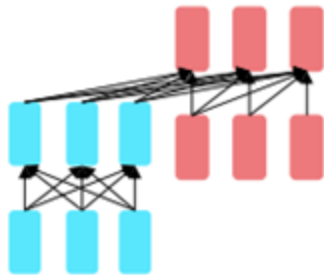
Decoders

- ❑ Simple left-to-right language models!
- ❑ Nice to generate from; can't condition on future words
- ❑ Examples: GPT-2, GPT-3, LaMDA



Encoders

- ❑ Gets bidirectional context – can condition on future!
- ❑ Masked language models
- ❑ Examples: BERT, RoBERTa

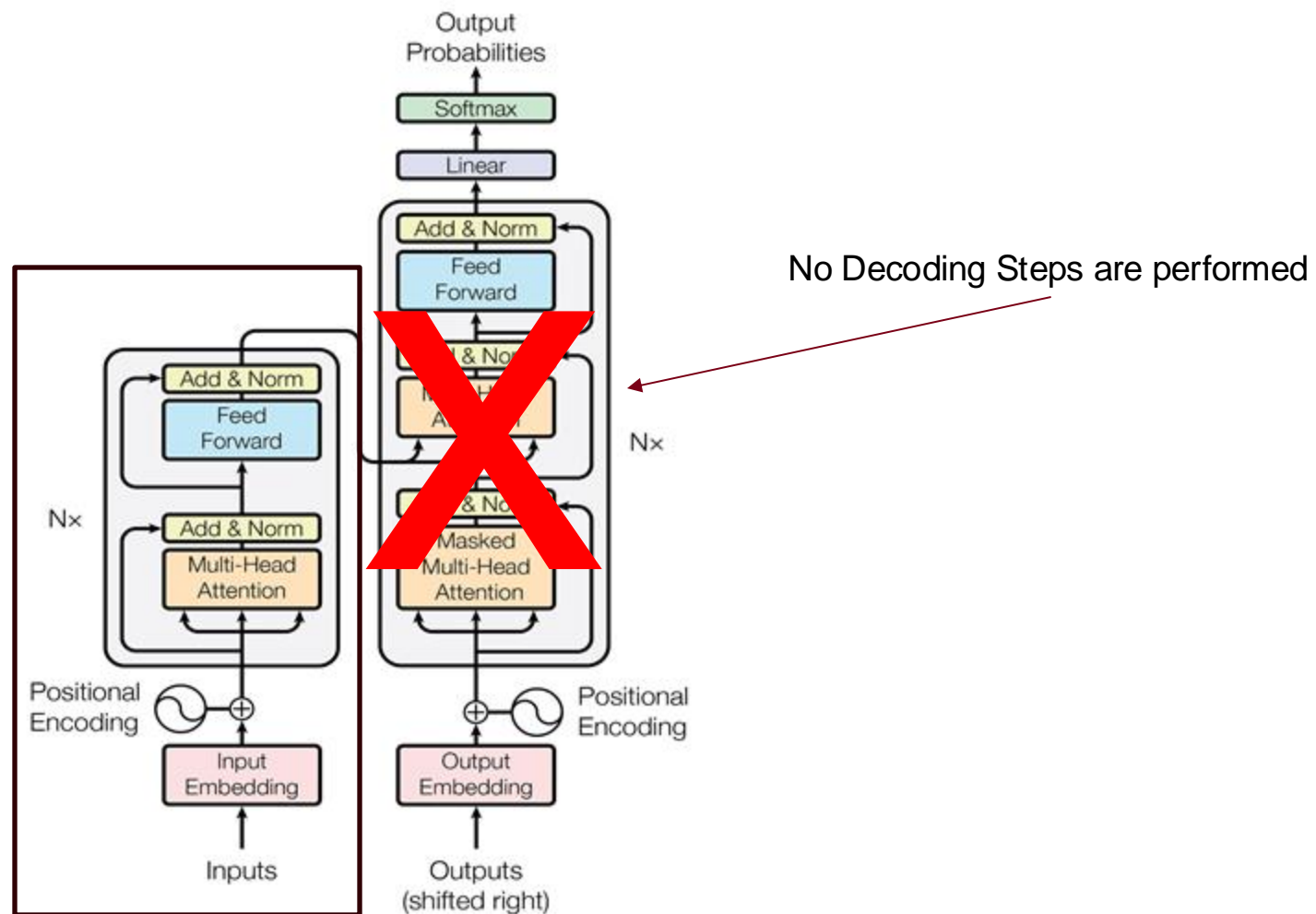


**Encoder-
Decoders**

- ❑ Good parts of decoders and encoders?
- ❑ What's the best way to pretrain them?
- ❑ Examples: T5, BART

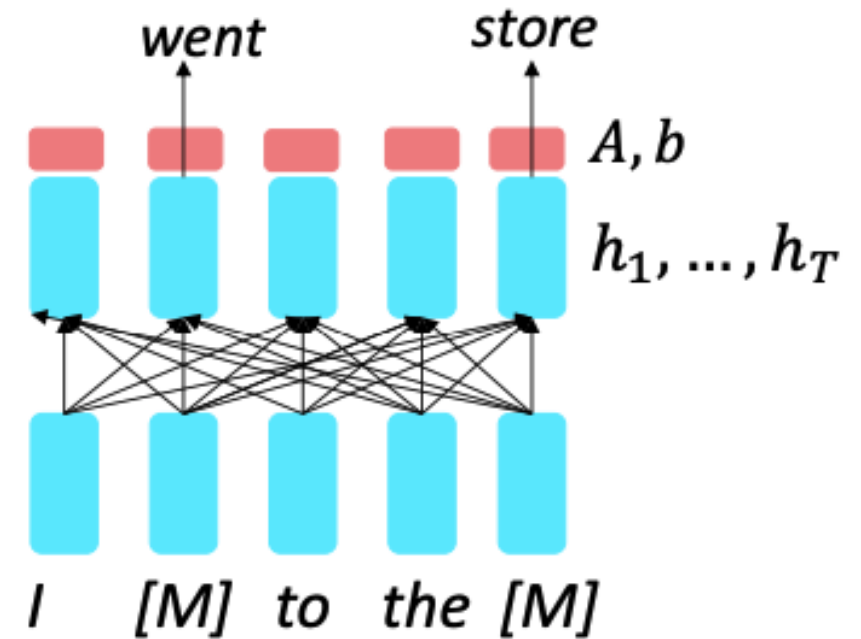


Encoder Only



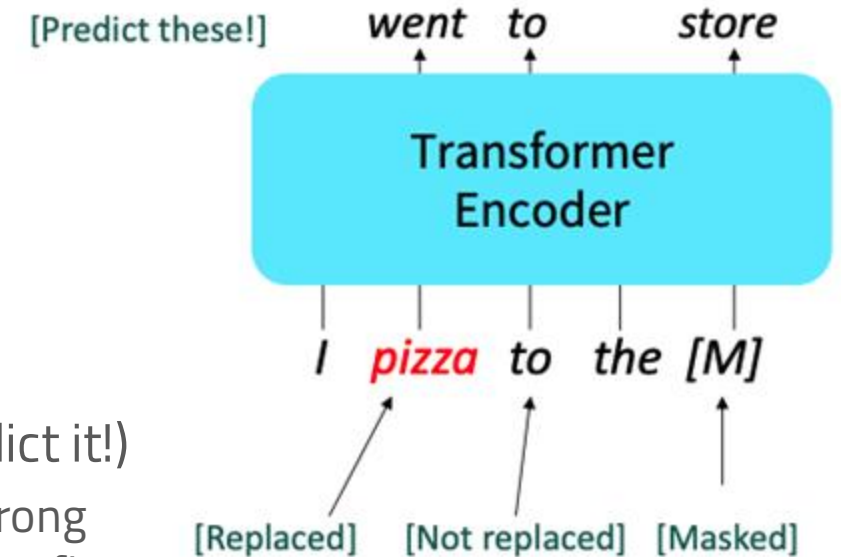
Pretraining and finetuning encoders

- ❑ So far, we've looked at language model pretraining. But, encoders get **bidirectional context**, so we can't do language modeling!
- ❑ Idea: replace some fraction of words in the input with a special **[MASK]** token; predict these words.
- ❑ Only add loss terms from words that are "masked out." If \hat{x} is the masked version of x we're learning $P_{\theta}(x | \hat{x})$ called Masked LM.



BERT: Bidirectional Encoder Representations from Transformers

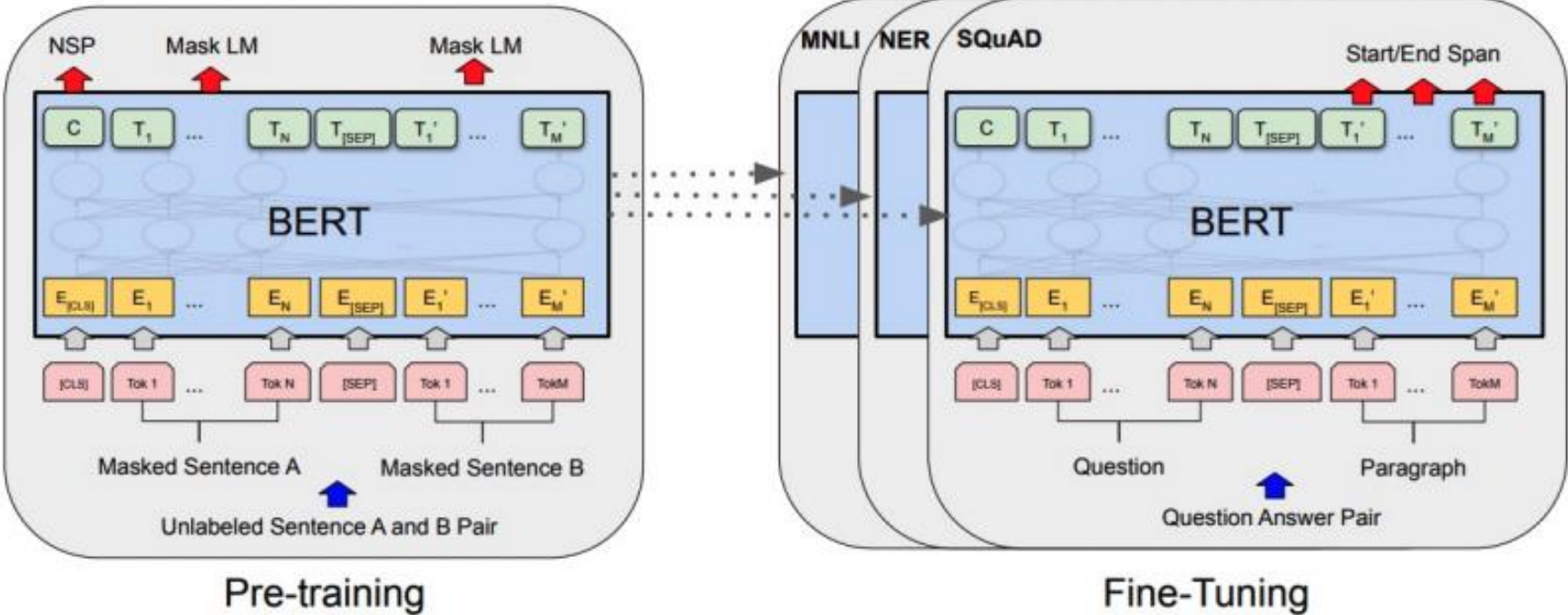
- ❑ Devlin et al., 2018 proposed the “Masked LM” objective and released the weights of their pretrained Transformer (BERT).
- ❑ Details about Masked LM for BERT:
 - Predict a random 15% of (sub)word tokens.
 - ✓ **Replace** input word with [MASK] 80% of the time
 - ✓ **Replace** input word with a random token 10% of the time
 - ✓ Leave input word unchanged 10% of the time (but still predict it!)
 - Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



Original text: "I went to the store"

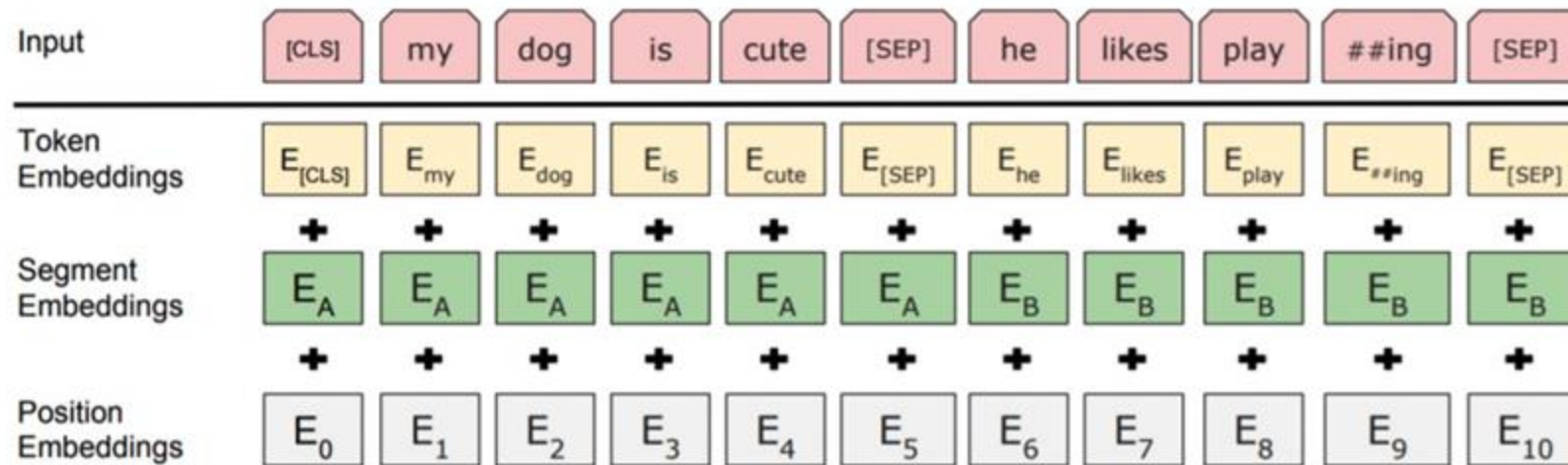
BERT: Bidirectional Encoder Representations from Transformers

Unified Architecture: As shown below, there are minimal differences between the pre-training architecture and the fine-tuned version for each downstream task



BERT: Bidirectional Encoder Representations from Transformers

□ The pretraining input to BERT was two separate contiguous chunks of text:



□ BERT was trained to predict whether one chunk follows the other or is randomly sampled.

- Later work; RoBERTa (Liu et al., 2019) has argued this “next sentence prediction” is not necessary.



Details about BERT Training

- ❑ Two models were released:
 - BERT-base: 12 layers, 768-dim hidden, 12 attention heads, 110 million params.
 - BERT-large: 24 layers, 1024-dim hidden, 16 attention heads, 340 million params.
- ❑ Trained on:
 - BookCorpus (800 million words)
 - English Wikipedia (2,500 million words)
- ❑ Pretraining is expensive and impractical on a single GPU.
 - BERT was pretrained with **64 TPU chips for a total of 4 days**
 - TPUs are special tensor operation acceleration hardware developed by Google
- ❑ Finetuning is practical and common on a single GPU
 - “Pretrain once, finetune many times.”



BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)

□ BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- **QQP**: Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

BERT-base was chosen to have the same number of parameters as OpenAI's GPT



BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)

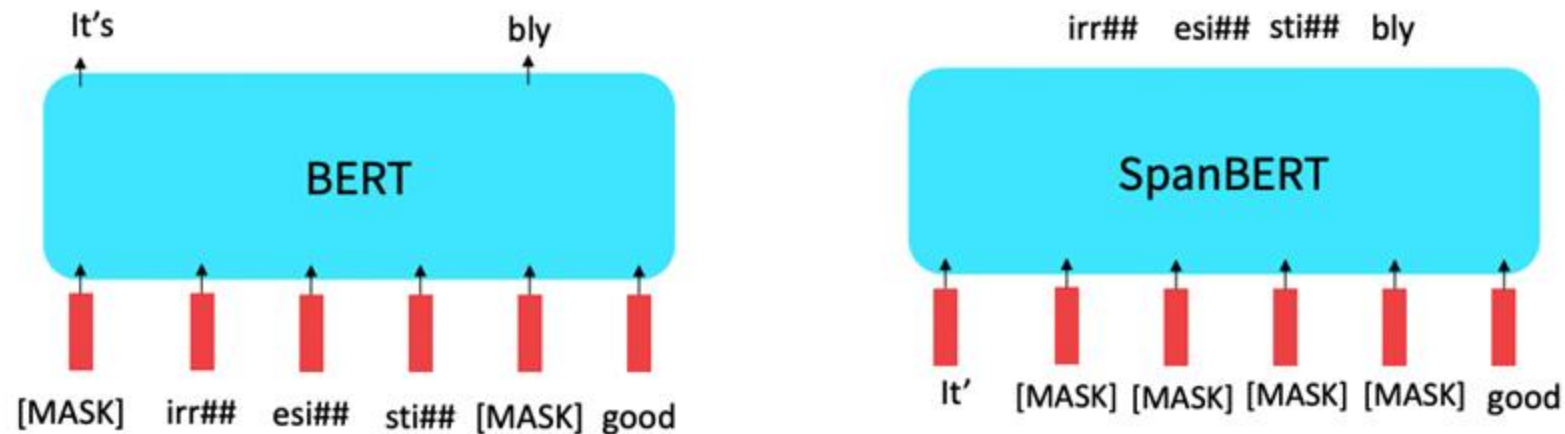
Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hrs)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x V100 (1 day)
GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100
GPT-3	96	12288	96	175B	694GB	?

Brown et al, "Language Models are Few-Shot Learners", arXiv 2020



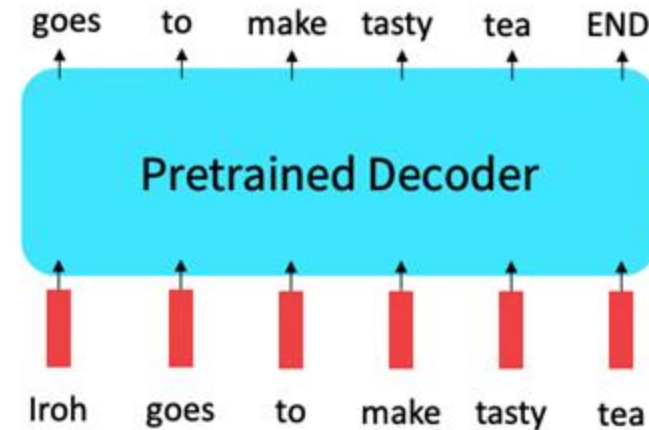
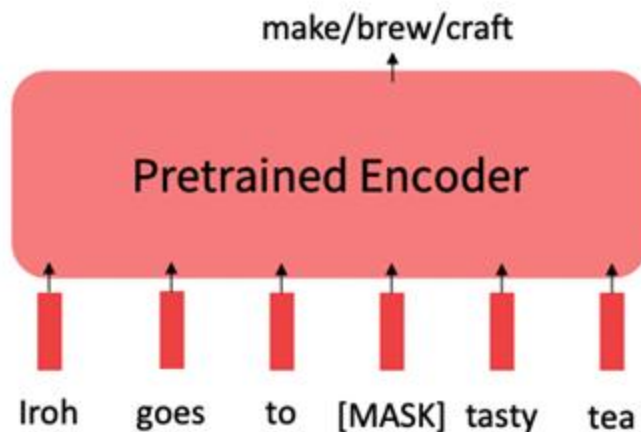
Extension of BERT

- You'll see a lot of BERT variants like RoBERTa, SpanBERT, ++
 - RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
 - SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task

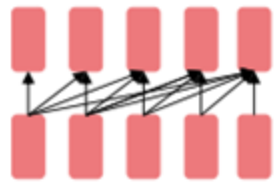


Limitations of pretrained encoders

- ❑ If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice **autoregressive** (1-word-at-a-time) generation methods.

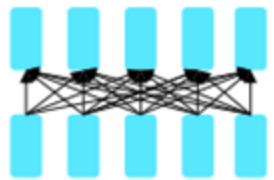


Pretraining for three types of architectures



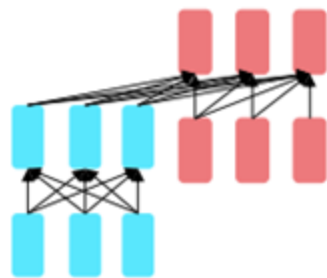
Decoders

- ❑ Simple left-to-right language models!
- ❑ Nice to generate from; can't condition on future words
- ❑ Examples: GPT-2, GPT-3, LaMDA



Encoders

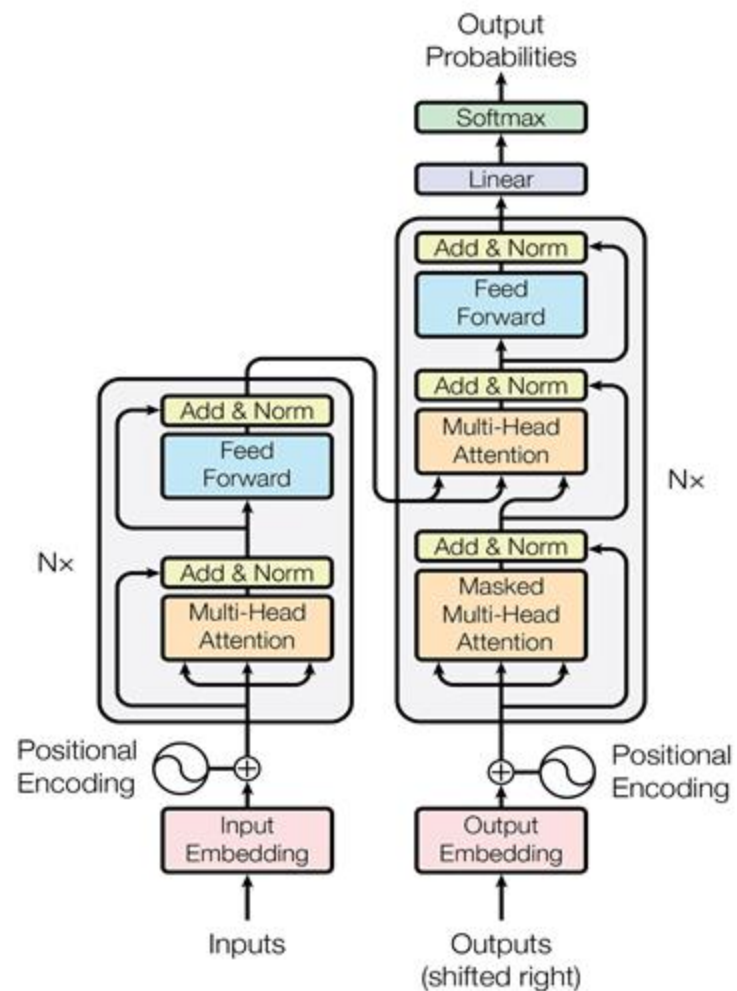
- ❑ Gets bidirectional context – can condition on future!
- ❑ Masked language models
- ❑ Examples: BERT, RoBERTa



**Encoder-
Decoders**

- ❑ Good parts of decoders and encoders?
- ❑ What's the best way to pretrain them?
- ❑ Examples: T5, BART

Encoder - Decoder

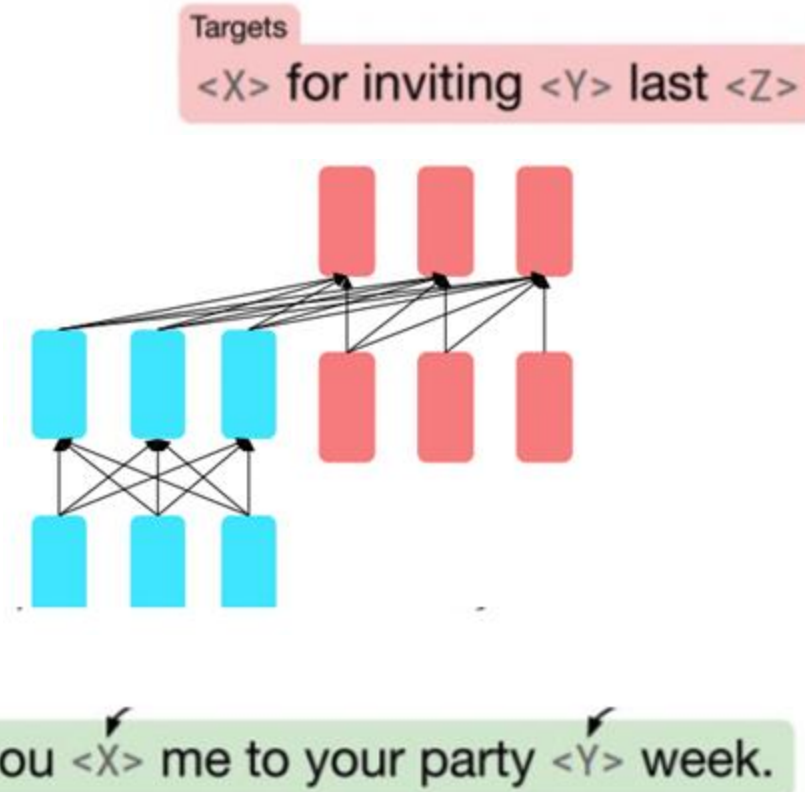


Pretraining encoder-decoders

- What Raffel et al., 2018 found to work best was **span corruption**. Their model: **T5**.
- Replace different-length spans from the input with unique placeholders ($\langle x \rangle$, $\langle y \rangle$); decode out the spans that were removed!

Original text

Thank you for inviting me to your party last week.



Pretraining encoder-decoders

- Raffel et al., 2018 found **encoder-decoders** to work better than decoders for their tasks, and span corruption (denoising) to work better than language modeling.

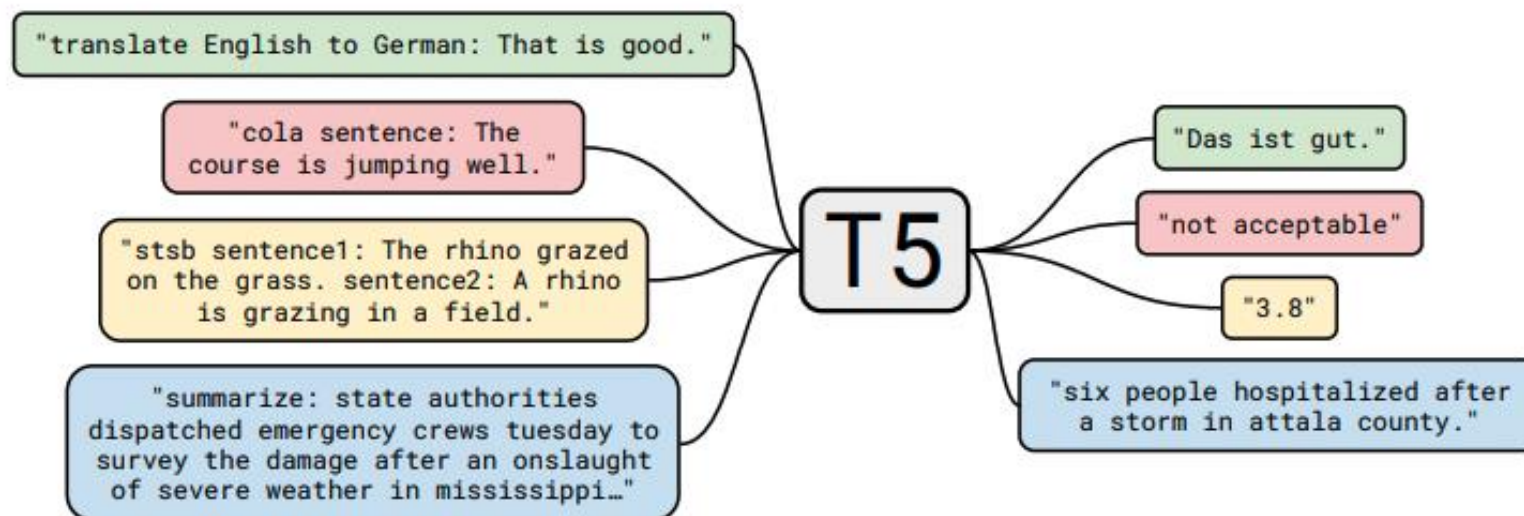
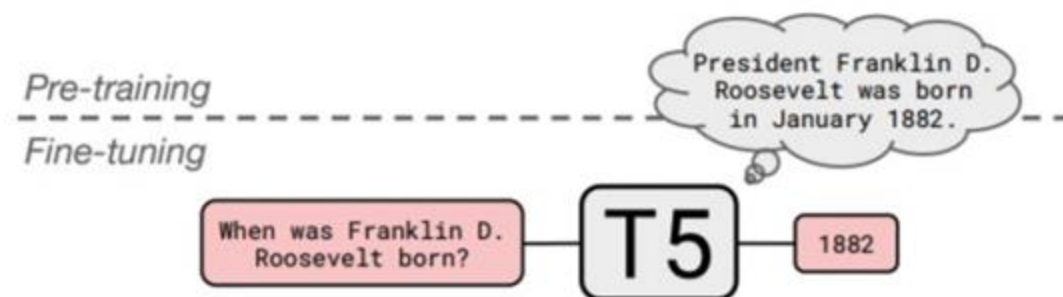
Architecture	Objective	Params	Cost	GLUE	CNN3M	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76



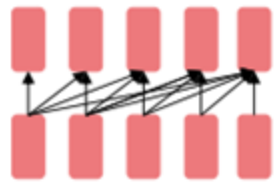
Pretraining encoder-decoders

A fascinating property of T5:

- finetune to answer a wide range of questions, retrieving knowledge from its parameters
- Multi-task learning

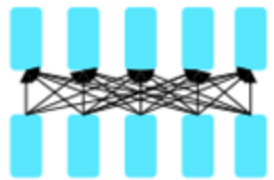


Pretraining for three types of architectures



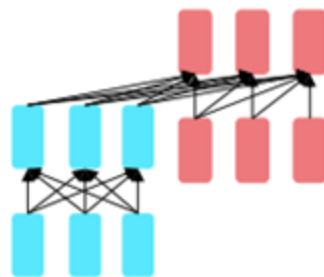
Decoders

- ❑ Simple left-to-right language models!
- ❑ Nice to generate from; can't condition on future words
- ❑ Examples: GPT-2, GPT-3, LaMDA



Encoders

- ❑ Gets bidirectional context – can condition on future!
- ❑ Masked language models
- ❑ Examples: BERT, RoBERTa



**Encoder-
Decoders**

- ❑ Good parts of decoders and encoders?
- ❑ What's the best way to pretrain them?
- ❑ Examples: T5, BART



GPT3, in-context learning, and VERY large language models



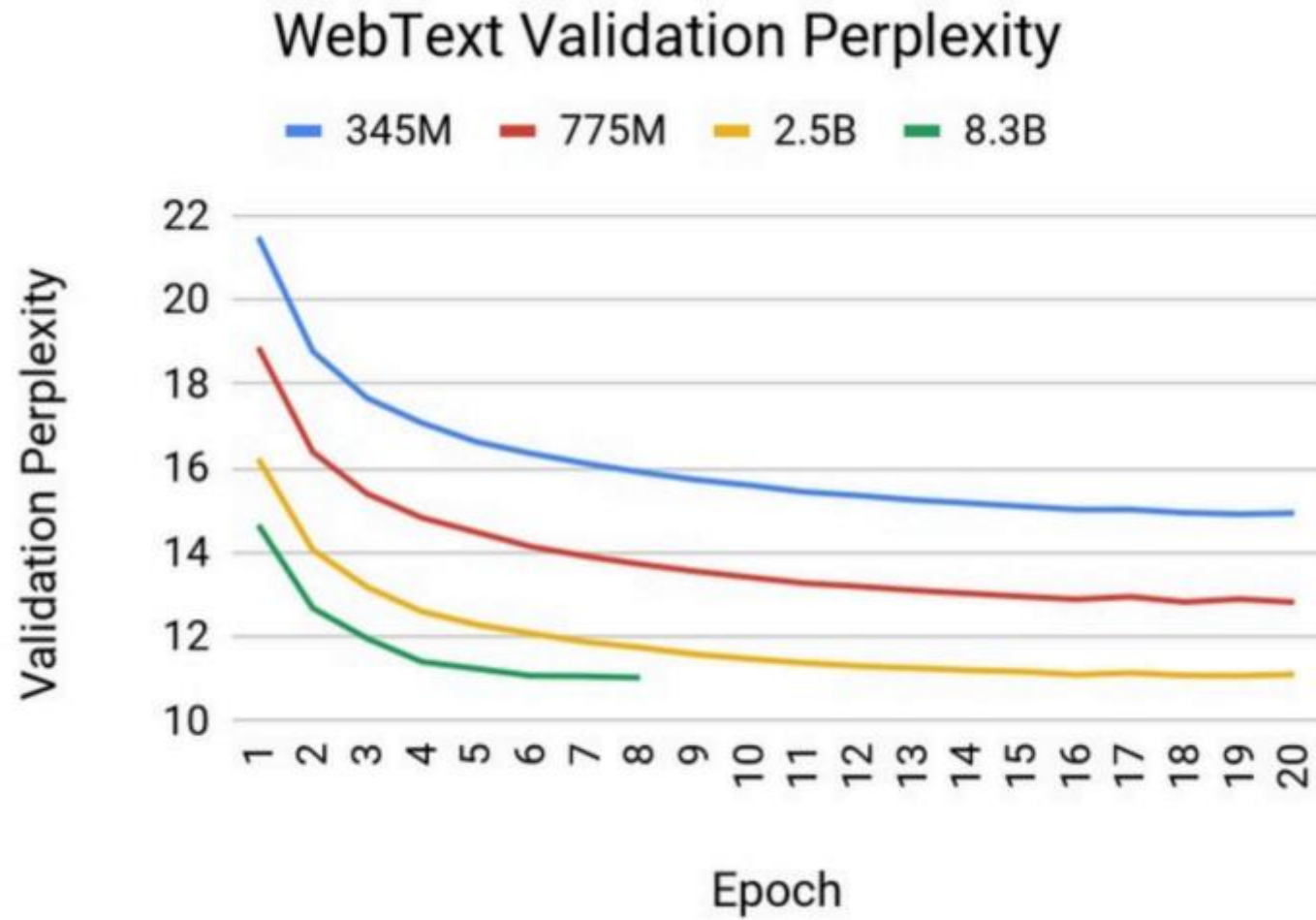
GPT3, in-context learning, and VERY large language models

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hrs)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x V100 (1 day)
GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100
GPT-3	96	12288	96	175B	694GB	?

Brown et al, "Language Models are Few-Shot Learners", arXiv 2020

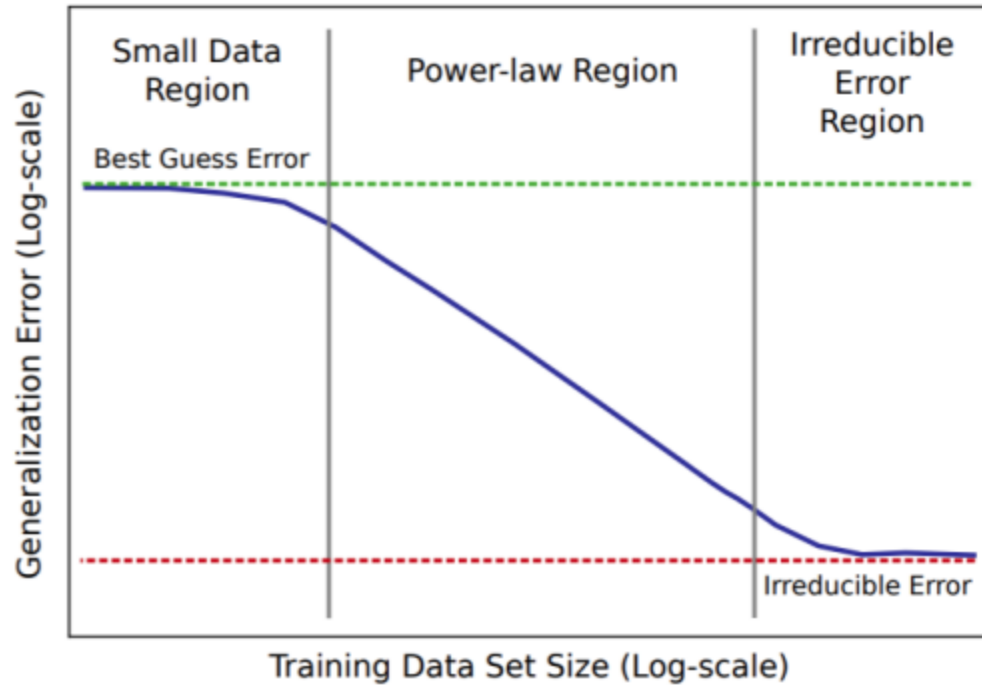


What are the scaling limits of large language models?

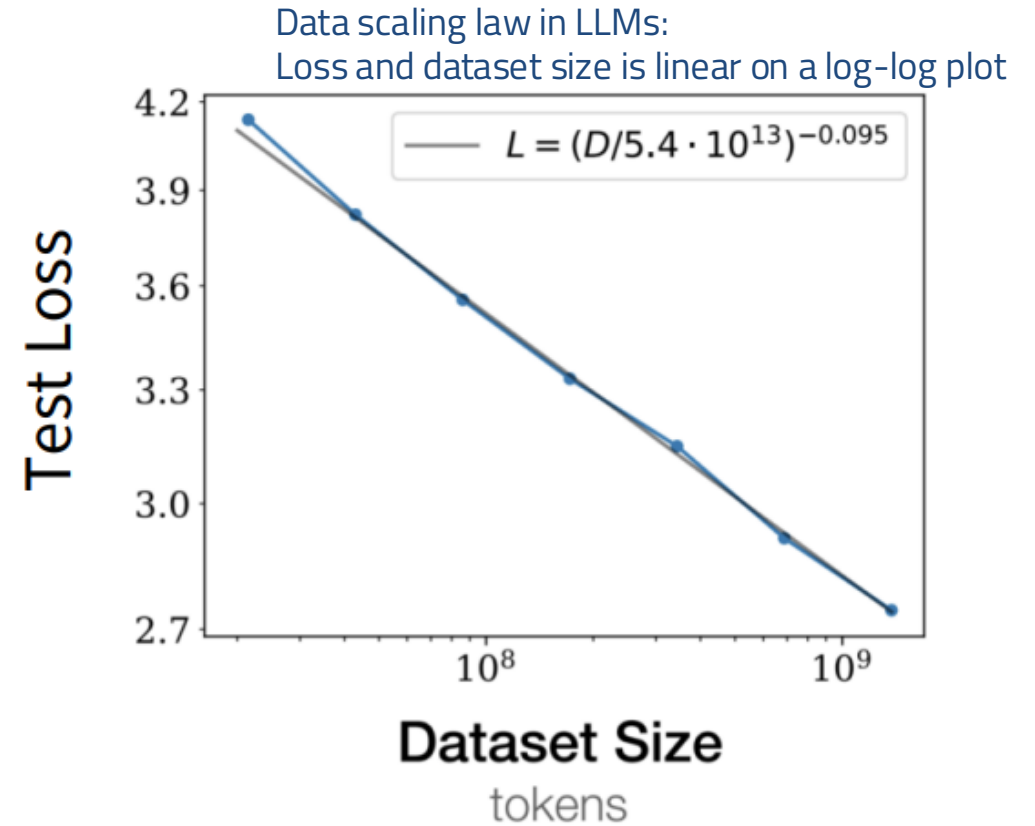


Data vs performance

- What's a data scaling law? simple formula that maps dataset size (n) to error



(Hestness+ 2017)



(Kaplan+ 2020)



GPT3

- GPT-2 but even larger: 1.3B -> 175B parameter models

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

- Trained on 570GB of Common Crawl
- 175B parameter model’s parameters alone take >400GB to store (4 bytes per param).
Trained in parallel on a “high bandwidth cluster provided by Microsoft”



GPT3, in-context learning, and VERY large language models

- ❑ So far, we've interacted with pretrained models in two ways:
 - Sample from the distributions they define
 - Fine-tune them on a task we care about, and then take their predictions
- ❑ **Emergent behavior:** Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide **within their contexts**.
 - GPT-3 is the canonical example of this. The largest T5 model had **11 billion** parameters. GPT-3 has **175 billion** parameters

GPT-2	48	1600	?	1.5B	40GB	
Megatron-LM	72	3072	32	8.3B	174GB	512x V100 (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100
GPT-3	96	12288	96	175B	694GB	?

Brown et al, "Language Models are Few-Shot Learners", arXiv 2020



In-context learning

- ❑ Step 1: Specify the task to be performed,
- ❑ Step 2: the conditional distribution (i.e., “loutre” ...) mimics performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

```
“  
thanks -> merci  
hello -> bonjour  
mint -> menthe  
otter ->  
”
```

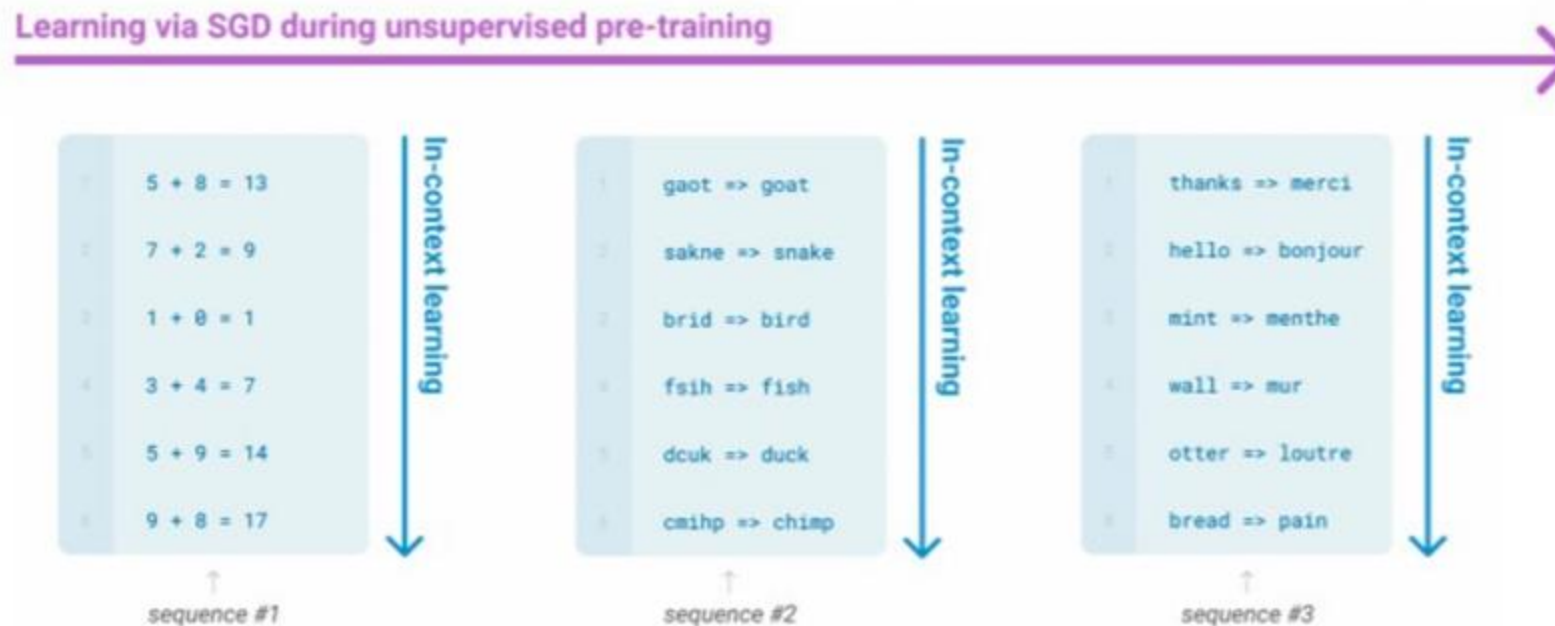
Output (conditional generation)

```
loutre ...
```



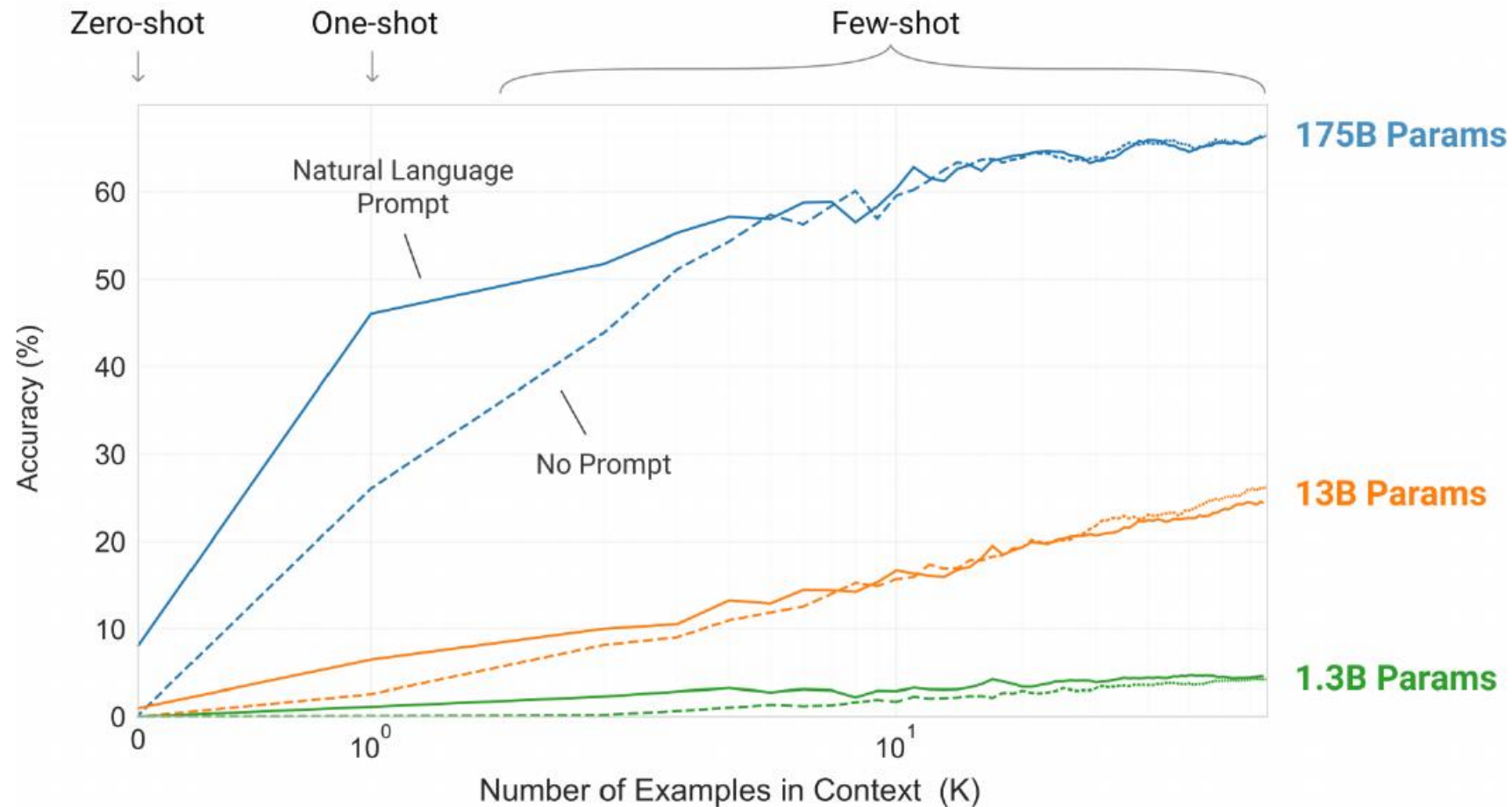
In-context learning

- Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.



GPT3

- **Key observation:**
 few-shot learning
 only works with the
 very largest models!



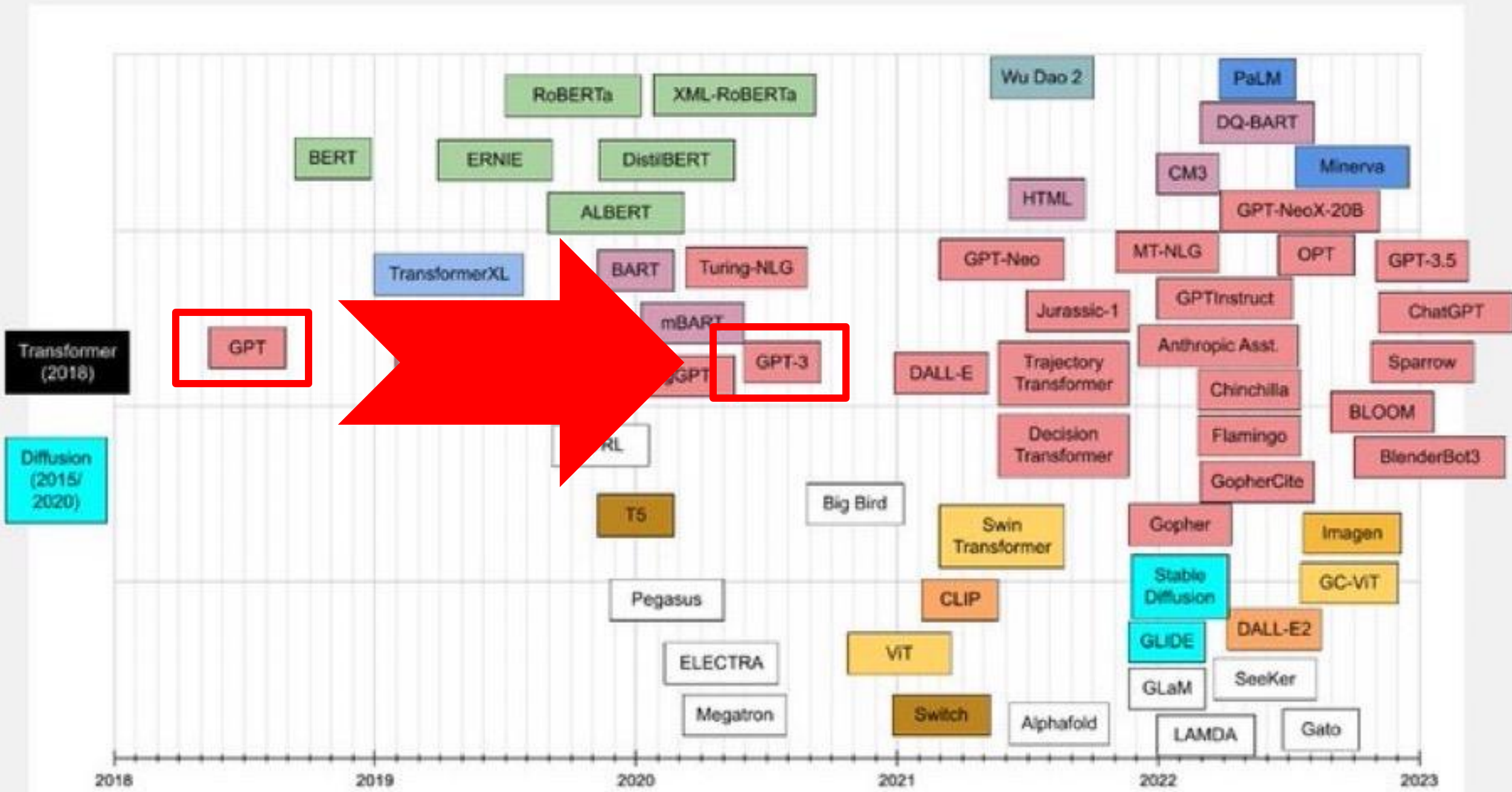
GPT3

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	89.0	91.0	96.9	93.9	94.8	92.5
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	76.1	93.8	62.3	88.2	92.5	93.3
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

- Sometimes very impressive, sometimes very bad
- Results on other datasets are equally mixed — but still strong for a few-shot model!





Scaling Laws



Scaling law in language model

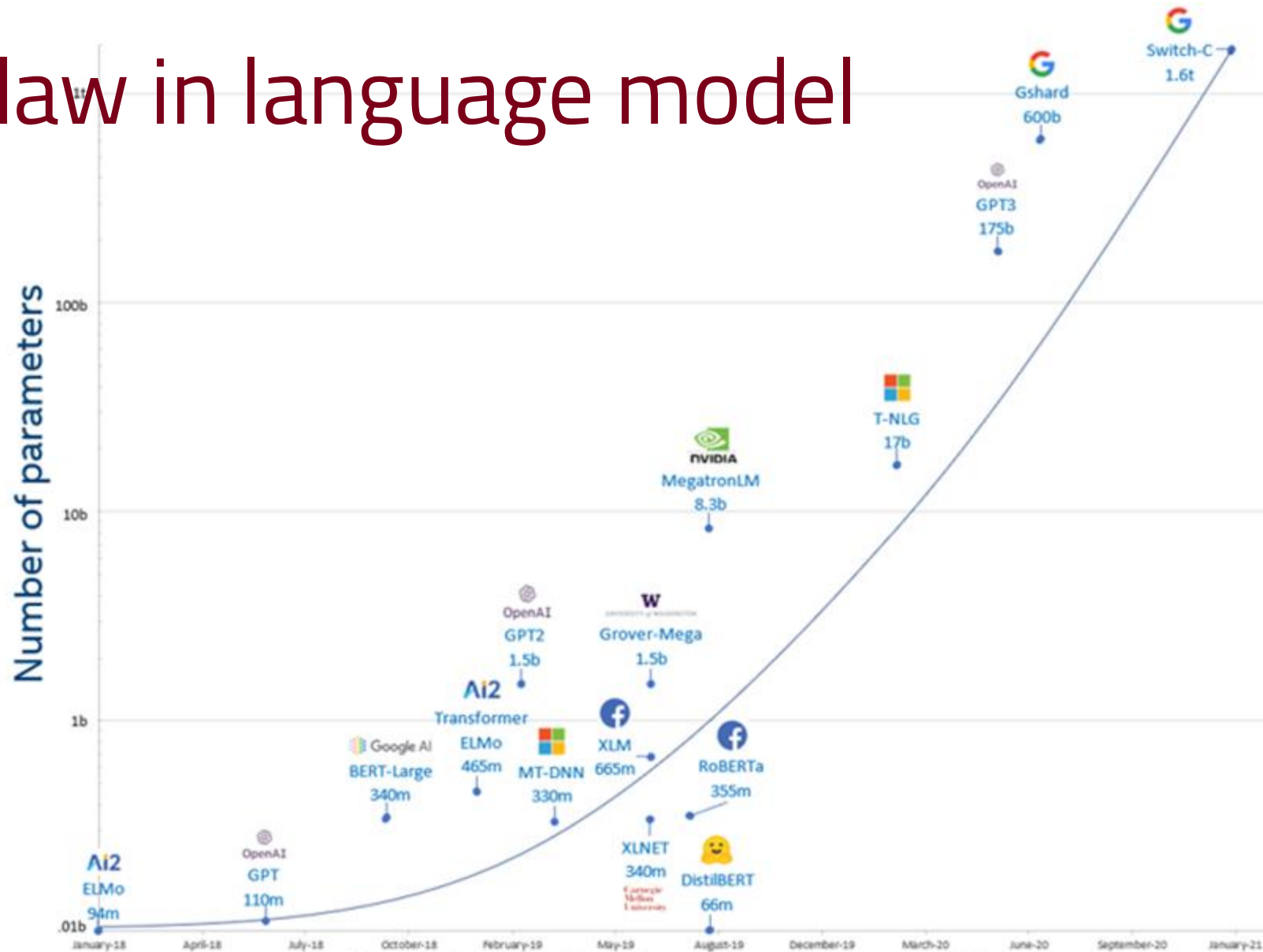


Figure 1: Exponential growth of number of parameters in DL models



Scaling Laws in LLM Pretraining

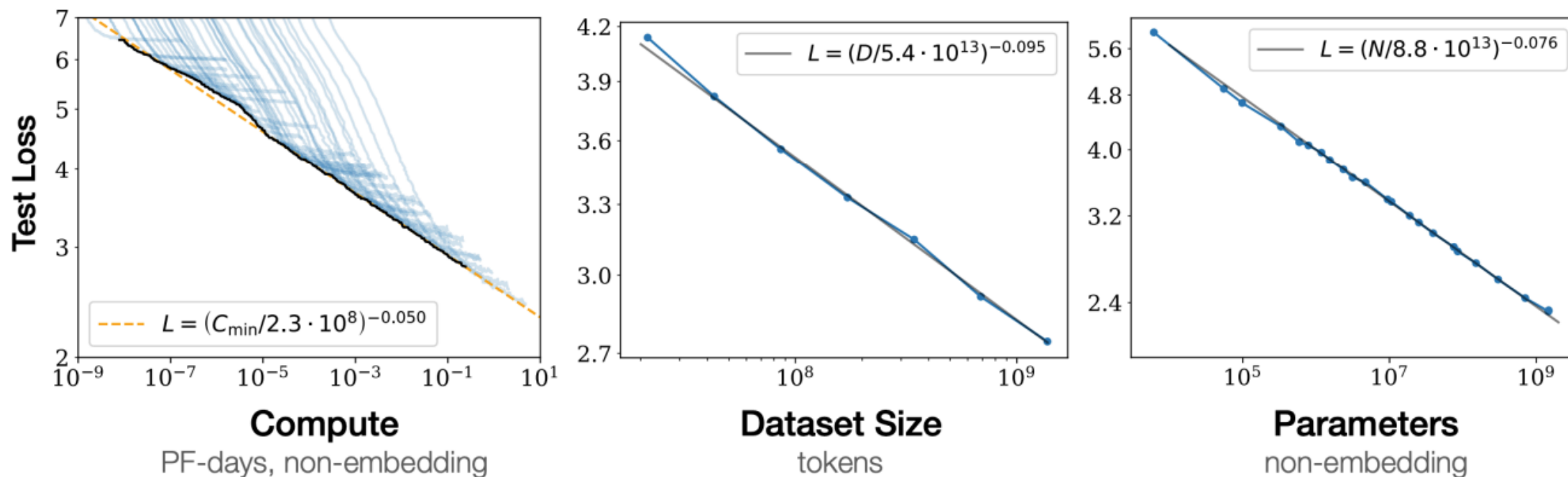


Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.



QUESTION ANSWERING

ARITHMETIC

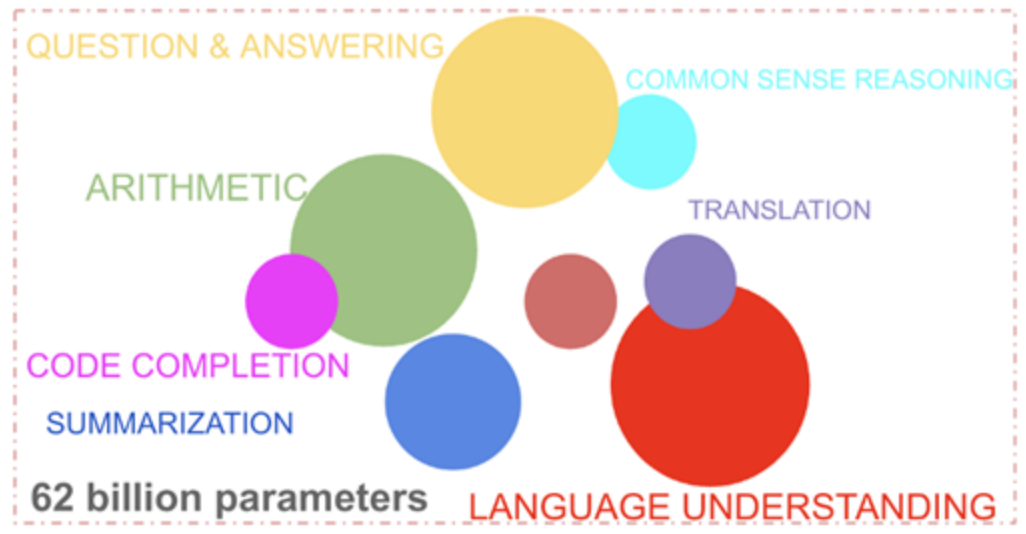
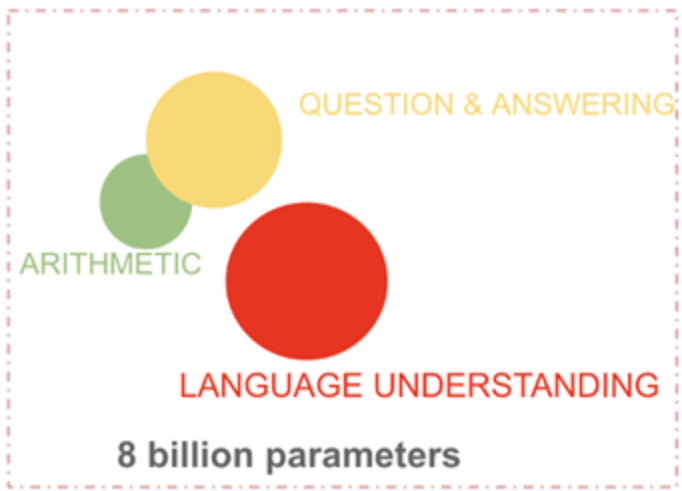


LANGUAGE UNDERSTANDING

8 billion parameters

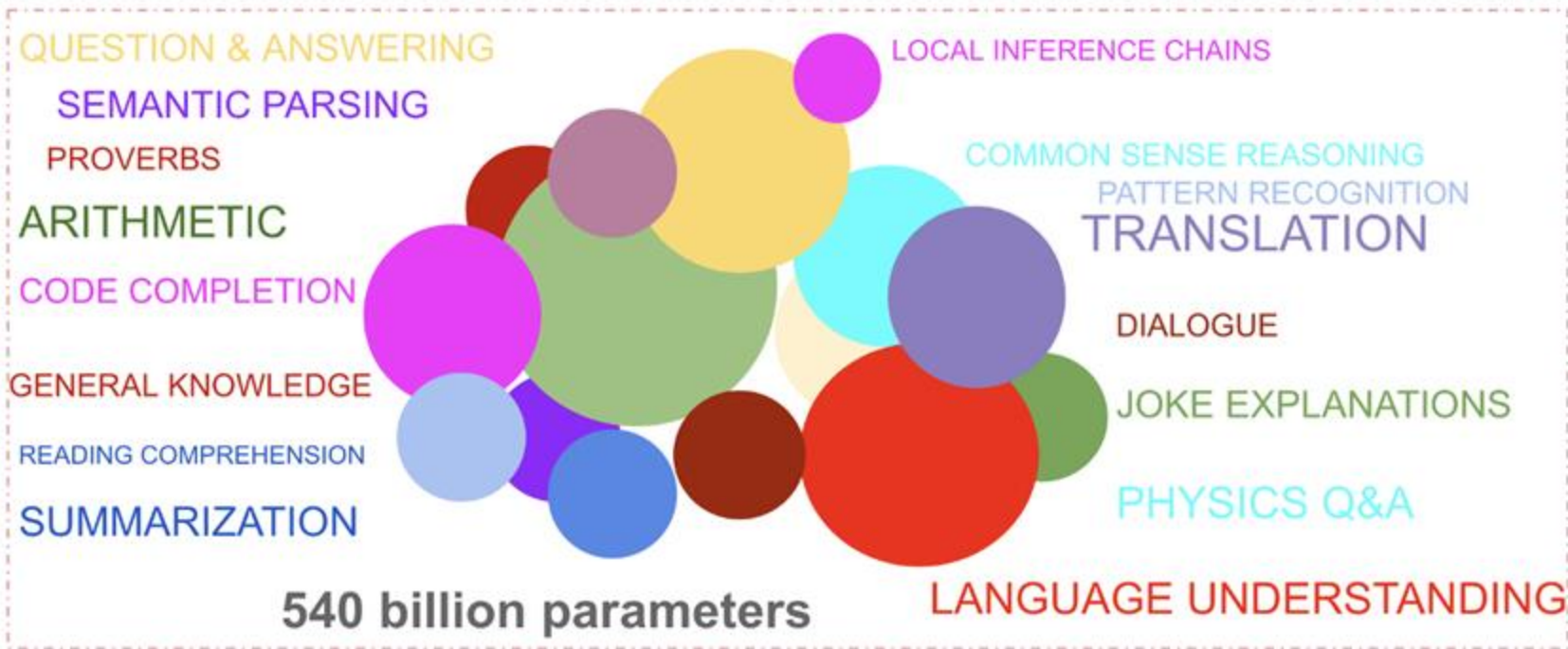
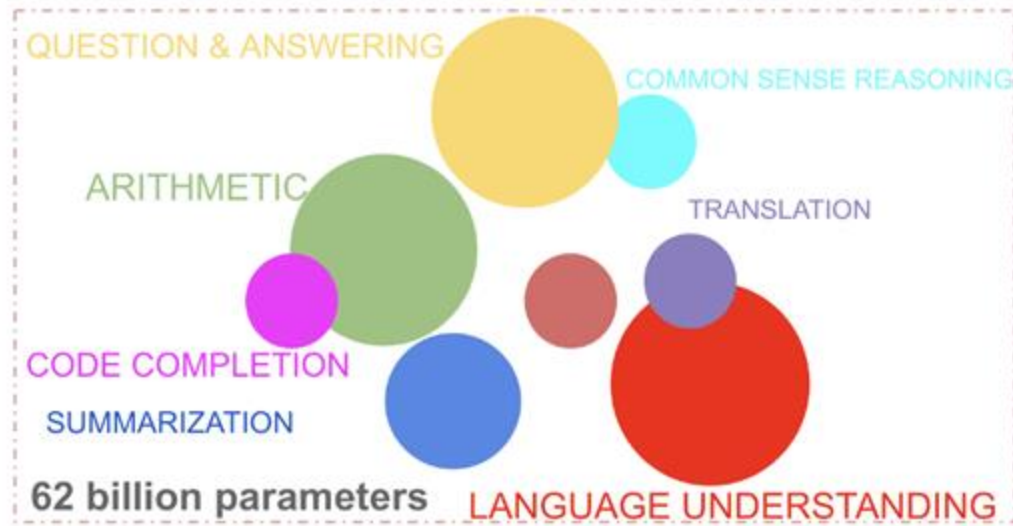
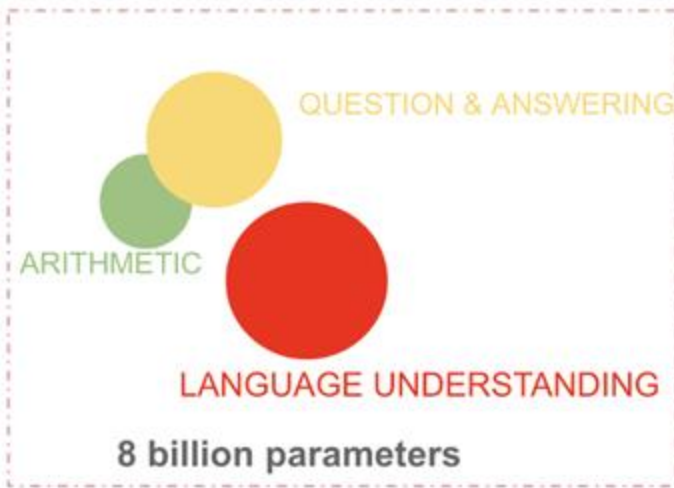
<https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html>





<https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html>



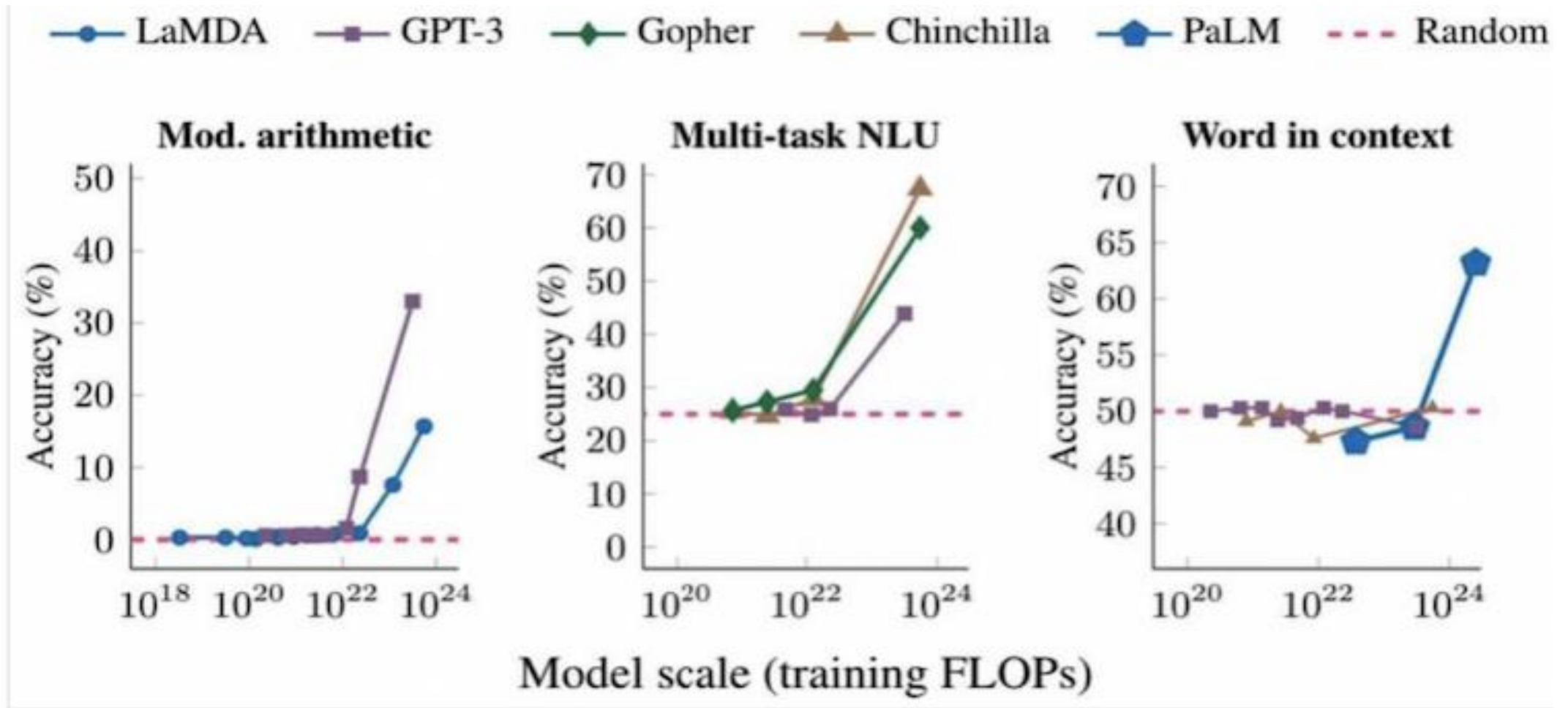


<https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html>



Emergent behavior from Scaling Law:

Quantum performance jump when +100B parameters

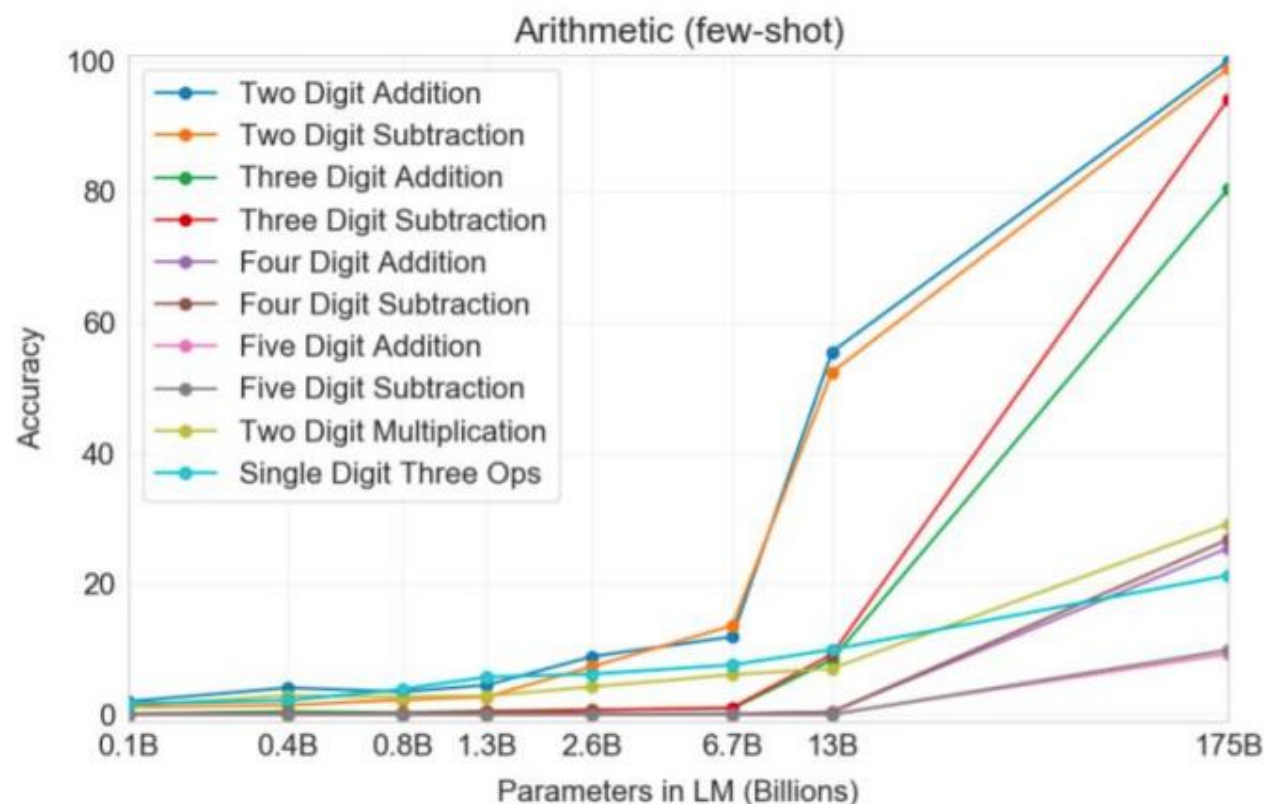


Jeff Dean <https://ai.googleblog.com/2023/01/google-research-2022-beyond-language.html>



Phase transitions

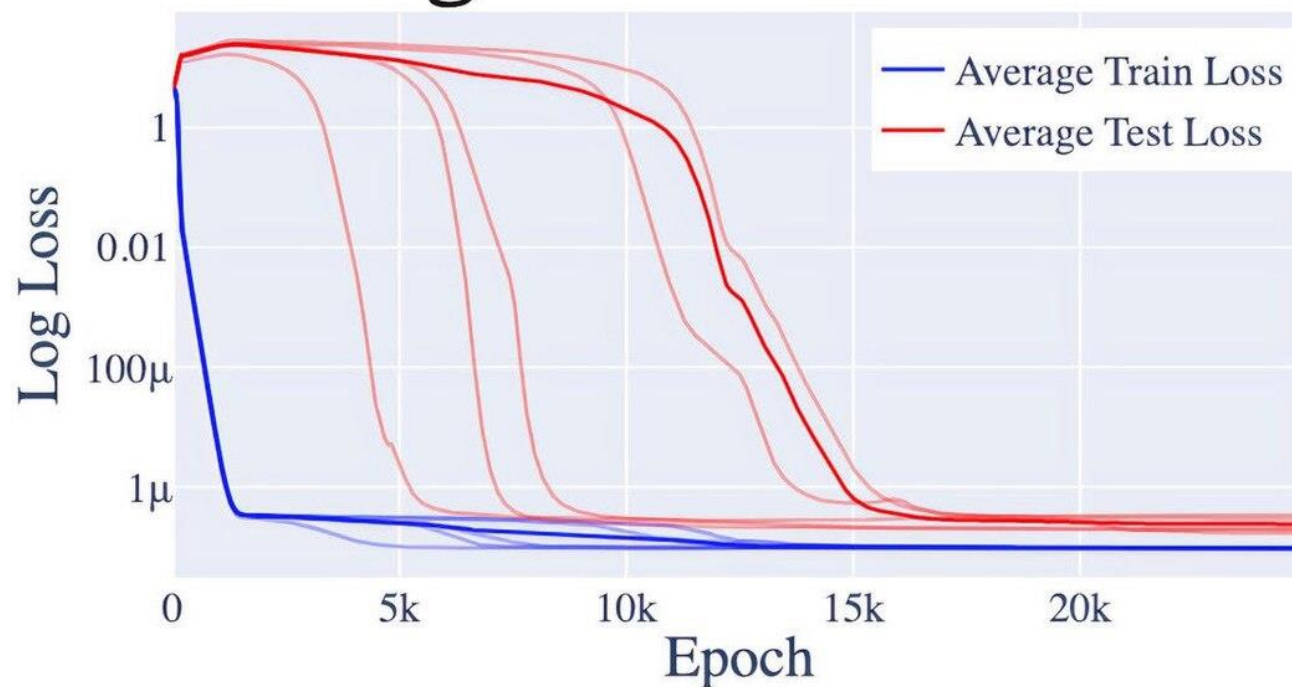
- Thus far: everything has had linear scaling (with different slopes).
- Phase transitions are sudden, discontinuous jumps in performance.
- The GPT-3 paper has some intriguing observations on phase transitions..
- Do we expect to see more phase transitions? This is probably the 'big unknown' in LM scaling!



Grokking

If we keep training past the point where our training loss declines – we can still see improvements in the test even past saturation of train loss towards 0

Grokking Modular Addition



Jeff Dean <https://ai.googleblog.com/2023/01/google-research-2022-beyond-language.html>



Scaling Law in Vision-Language Model



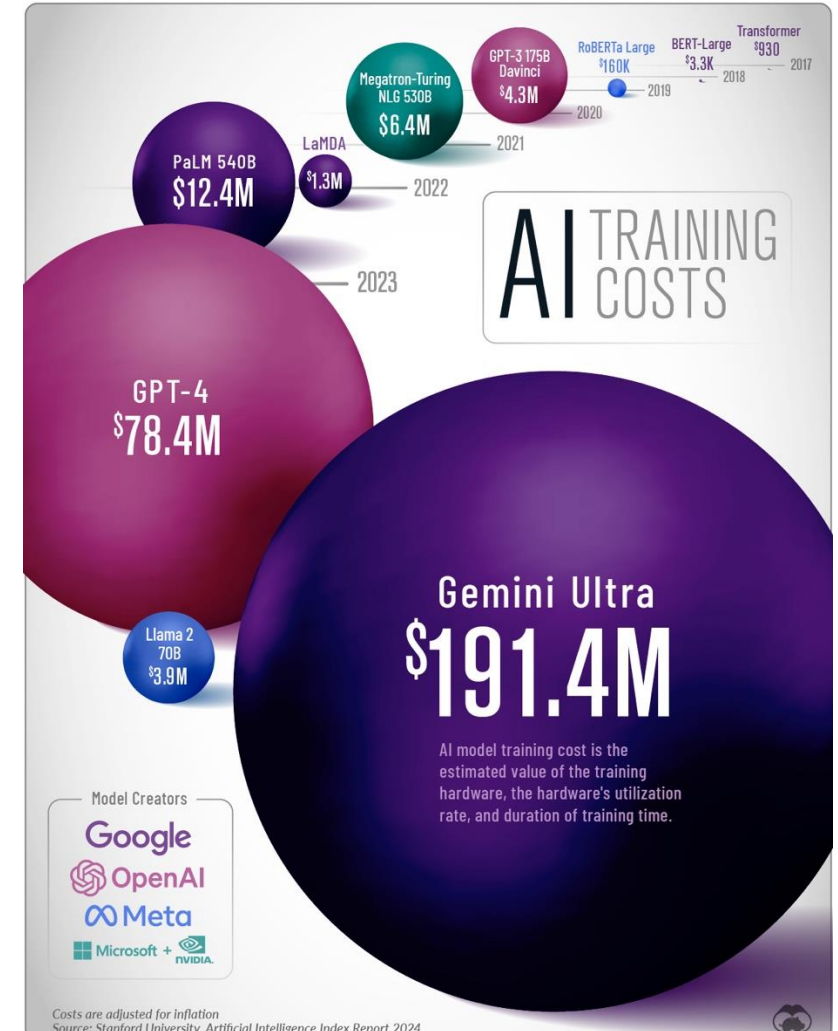
Figure 4. The generated image for the text "A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!". Note the model gets the text in the image "welcome friends" correct at 20B.

<https://towardsdatascience.com/a-quiet-shift-in-the-nlp-ecosystem-84672b8ec7af>

Pre-Training Costs Have Grown Over Time

Cost of Training at time of training (\$)

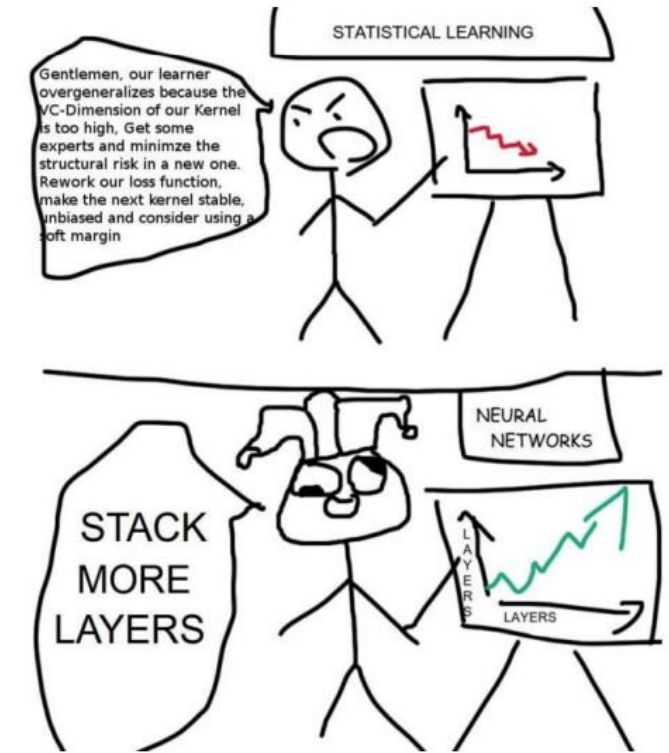
- ❑ BERT-Large (3.3k)
- ❑ GPT3 (4.3M)
- ❑ PaLM (12.4M)
- ❑ Llama2 (3.9M)
- ❑ GPT4 (78.4M)
- ❑ Gemini Ultra (191.43M)



Q: Can big language models solve every problem?

- ❑ We can use scaling laws to answer this!
 - For each capability (e.g. question answering)..
 - Build a scaling law for compute capacity.
 - Extrapolate the scaling curve.

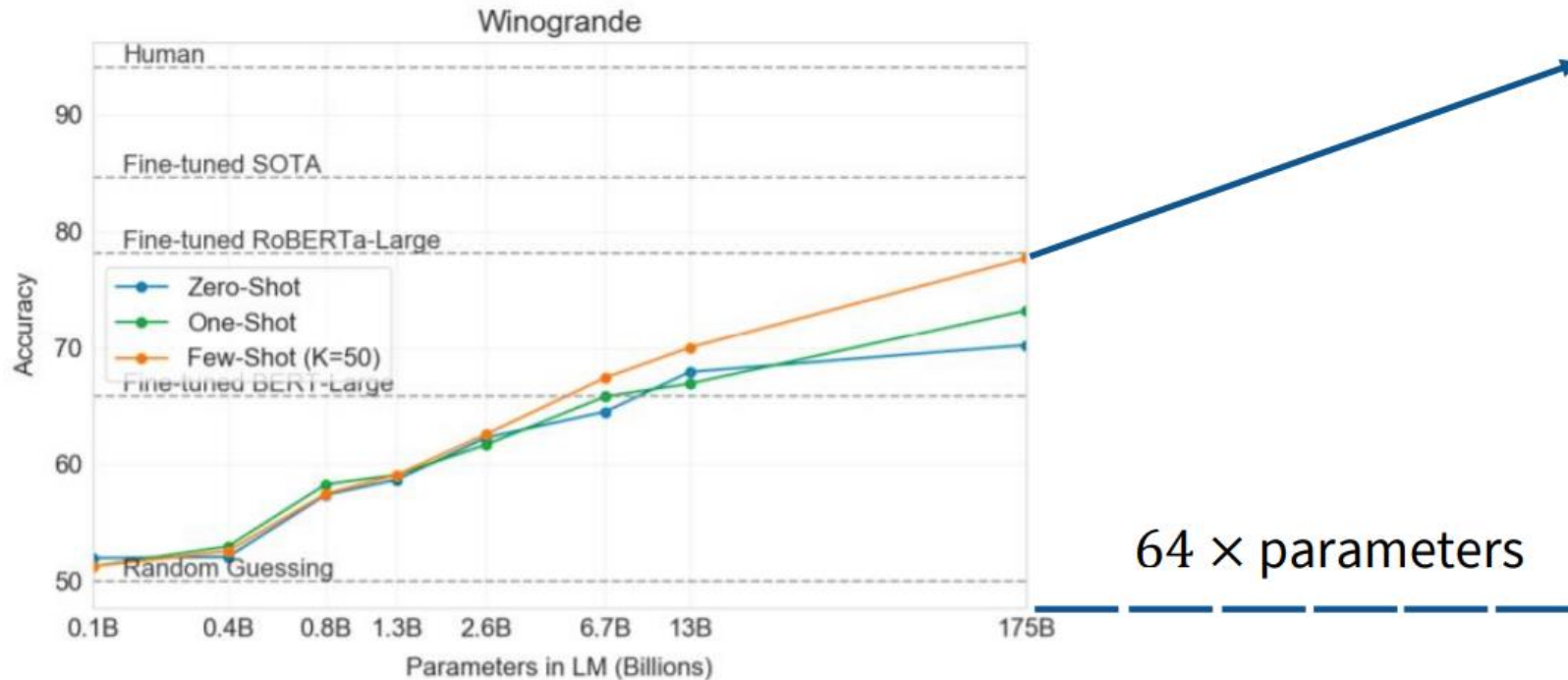
- ❑ Can 'reasonable' amounts of compute solve our problems?



Taken from r/programmerhumor



Human Level Reasoning Through Scaling?



If the scaling law holds.. Roughly 64 times more parameters will get us to human-level

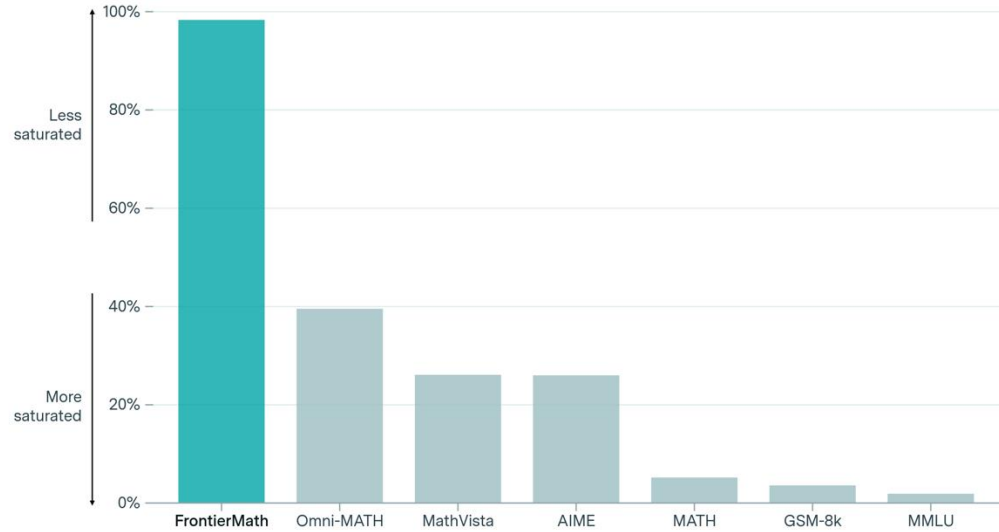


PhD Level AI?

FrontierMath vs. other mathematics benchmarks

EPOCH AI

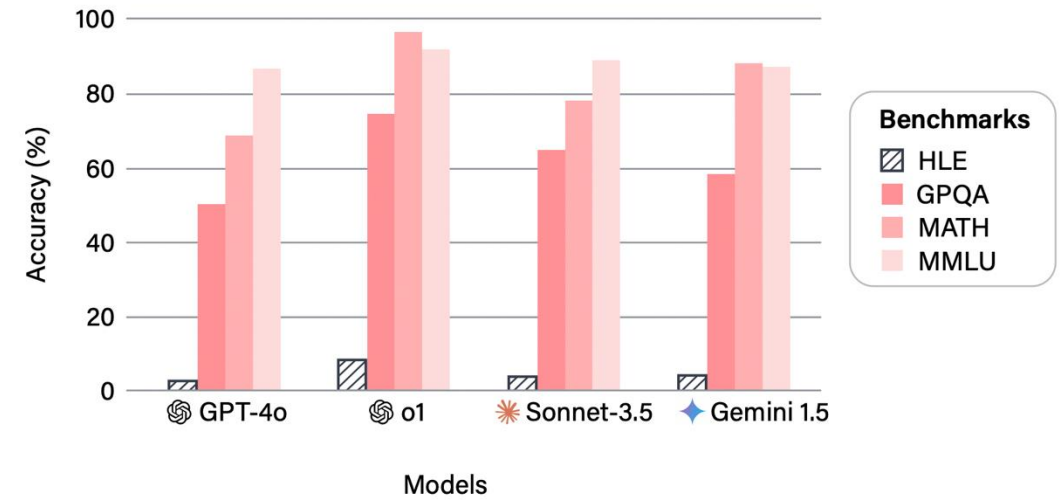
Problems not solved by leading AI models



CC-BY

epoch.ai

Accuracy of LLMs Across Benchmarks

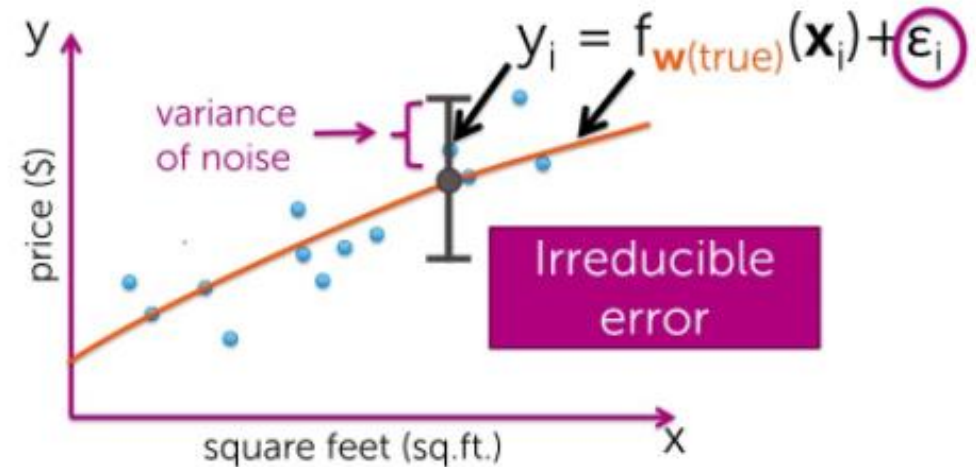


End of Pretrain-Scaling and the Limits of Next Token Prediction (NTP)



Potential Limitations: Irreducible Error

- ❑ Every dataset has some error due to randomness
- ❑ We don't have access to the generating processes of text (or in multimodal cases, vision/speech) and so cannot remove randomness
- ❑ Sets lower bound on our loss function



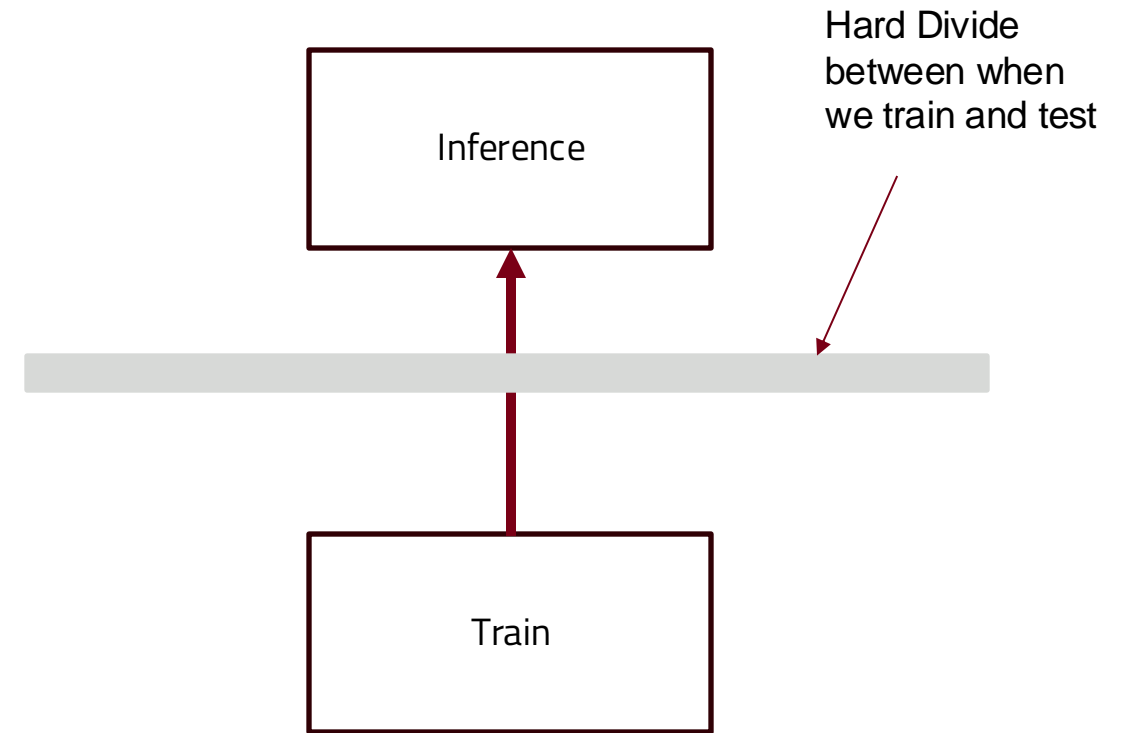
Potential Limitations: Embodied Understanding

- ❑ *Verbal Reasoning vs. Spatial Reasoning*
- ❑ *Crystallized Intelligence vs. Fluid Intelligence*
- ❑ Models that only predict next tokens may be missing out on the problem solving abilities more present in *Fluid Intelligence* and *Spatial Reasoning*



Potential Limitations: Train-Test Barrier

- ❑ The model does not update its parameters online as a person does
- ❑ The only way to 'learn' is to utilize better in-context-learning methods
- ❑ This barrier can prevent models from incorporating new information about the world
- ❑ If in-context learning is insufficient, we might get stuck



Potential Limitations: Domain Expertise

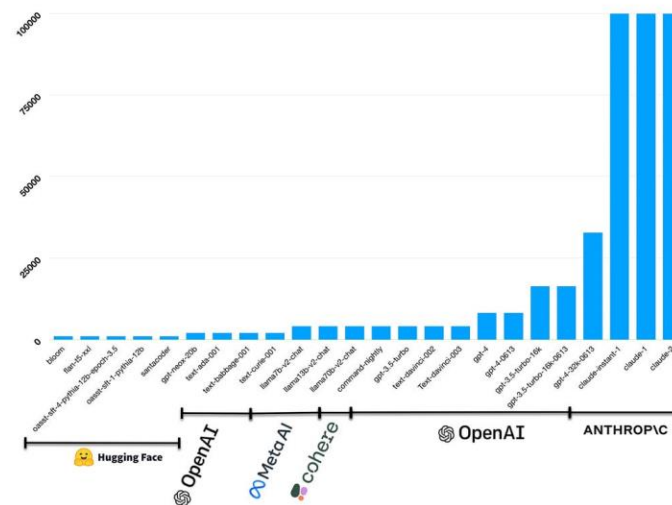
- ❑ When there is a limited number of tokens to train on, the performance of models is worse with next token prediction methods alone
- ❑ Low Resource Languages
- ❑ Underused APIs/Codebases



Potential Limitations: Long Context

- When the next token requires an extremely long context, LLM performance can suffer
- Current approaches have to increase context size or use some efficient methods to reduce computation/storage requirements

Large Language Model Context Size

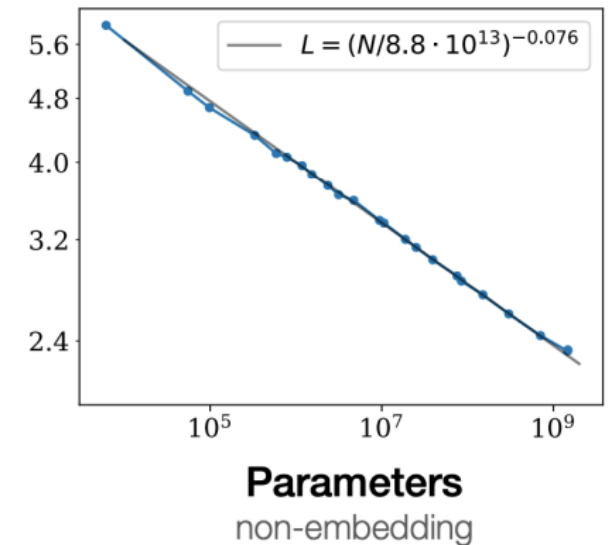
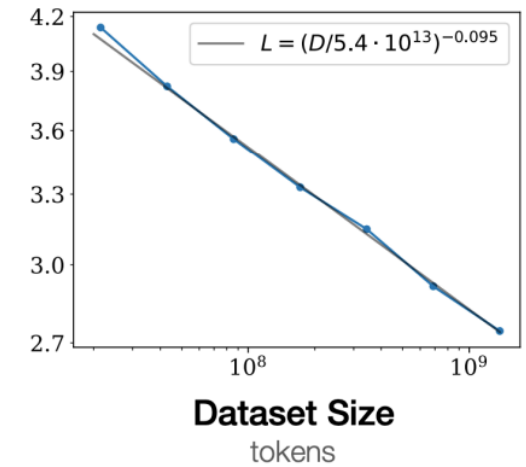
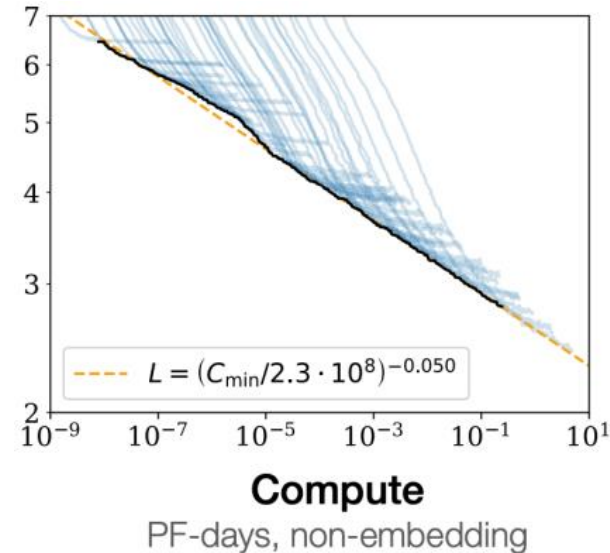


www.cobusgreyling.com



End of Pretrain Scaling

- ❑ Performance scales logarithmically with respect to inputs (dataset size/model size/training compute)
- ❑ Logarithmic improvements do not scale well
- ❑ Eventually, inputs become too expensive to continue to see improvements justified



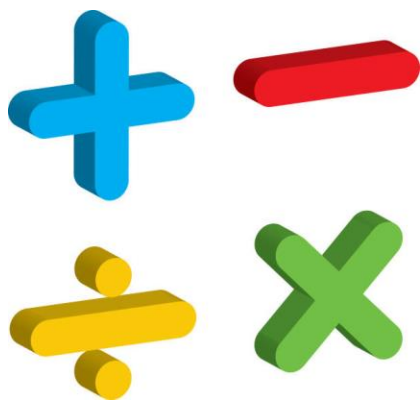
End of Pretrain Scaling: Bottlenecks



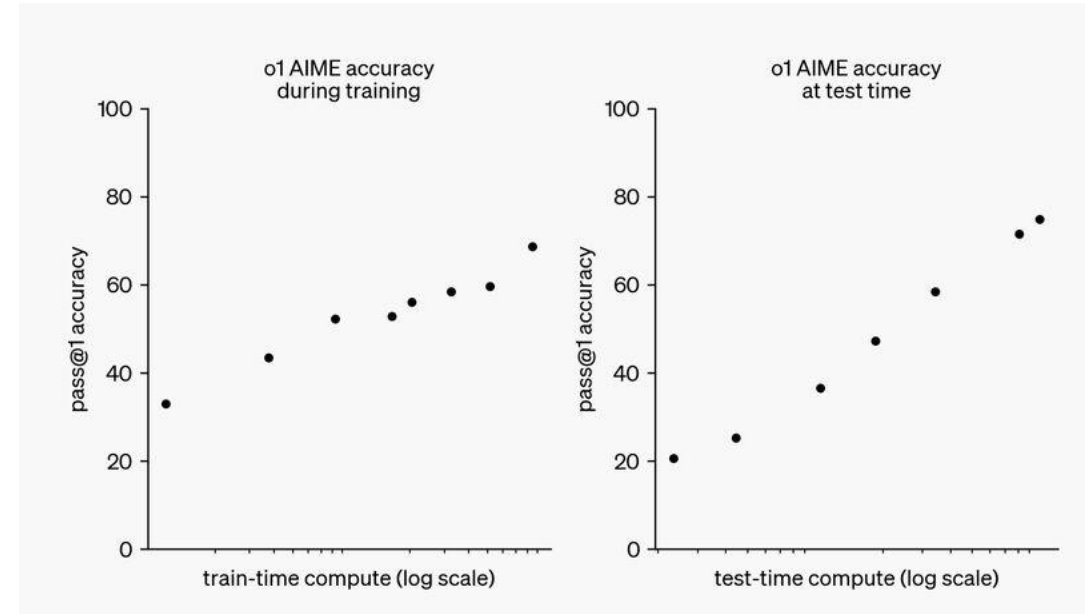
DATA ANNOTATIONS



Alternatives: Better Post-Training



Alternatives: Inference Scaling Laws



Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws

Large Language Monkeys: Scaling Inference Compute with Repeated Sampling.

- DeepSeek-Coder increases from 15.9% with one sample to 56% on SWE-Bench

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters.

- PaLM 2-S beats a 14x larger model on MATH with test-time search.

Image credits: Jim Fan



Alternatives: Reasoning

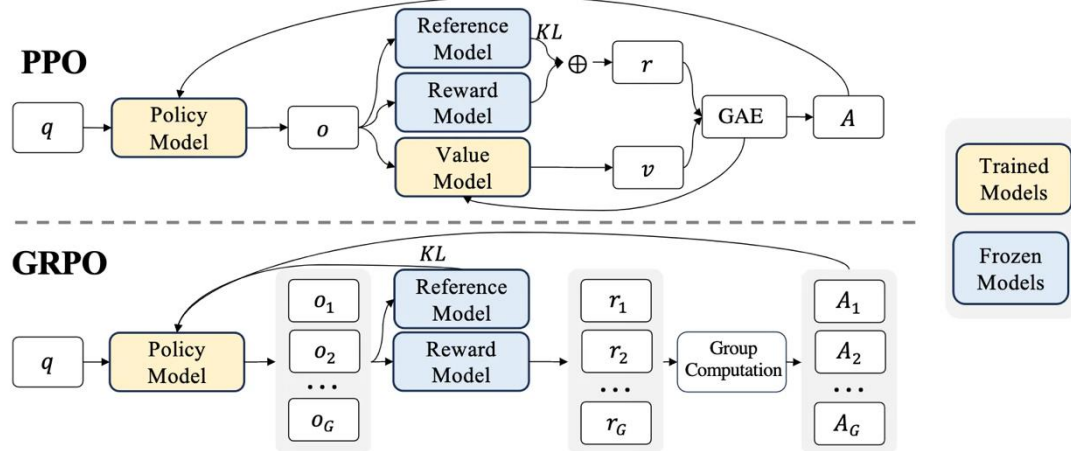
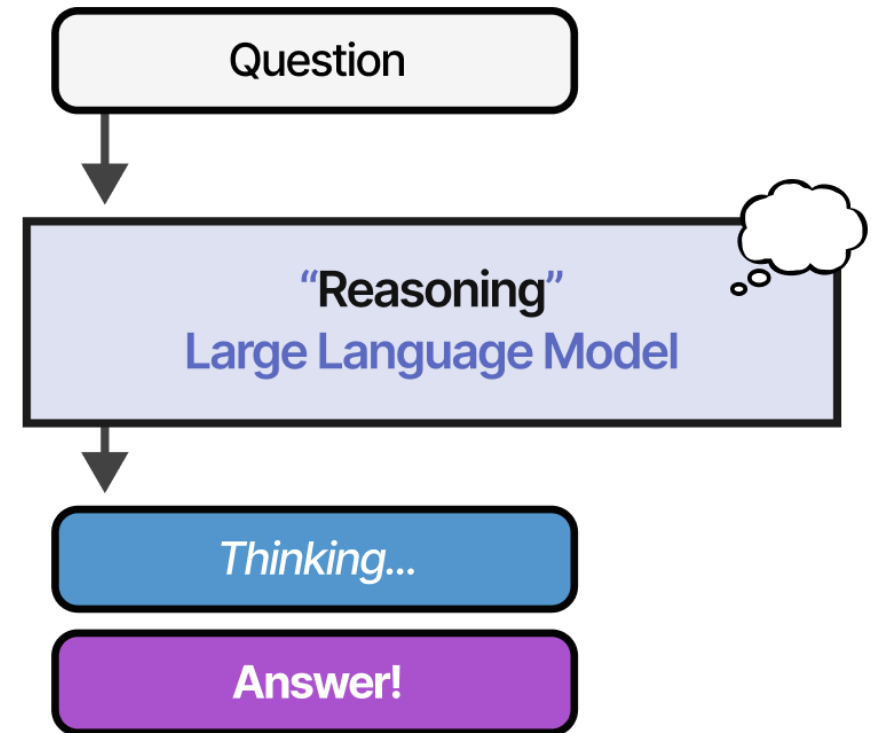
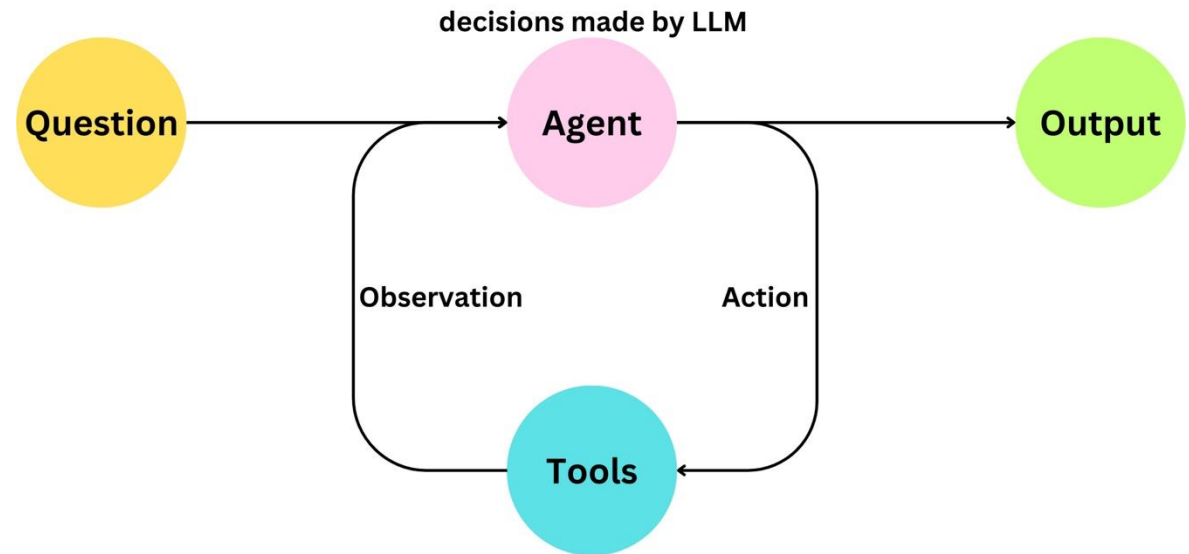
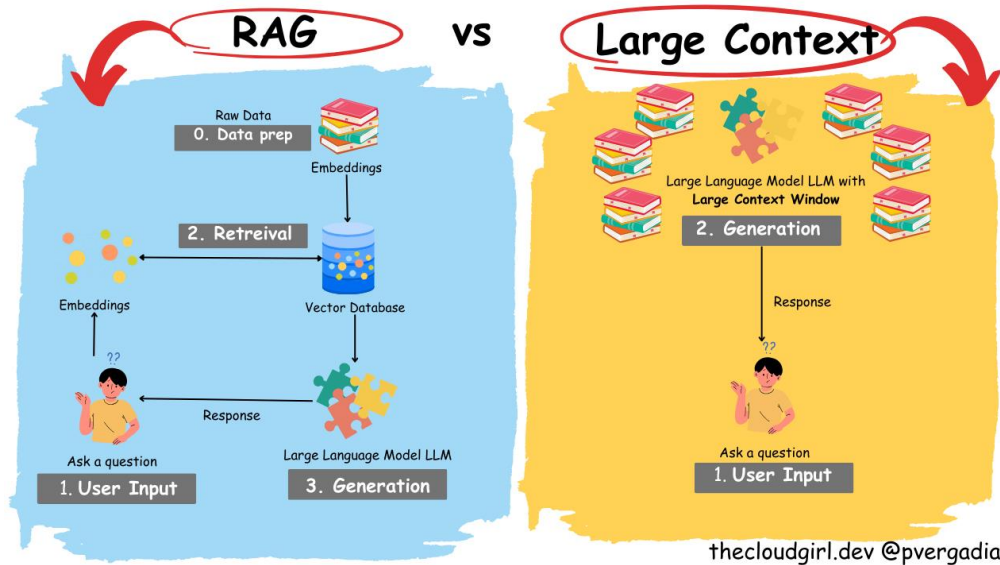


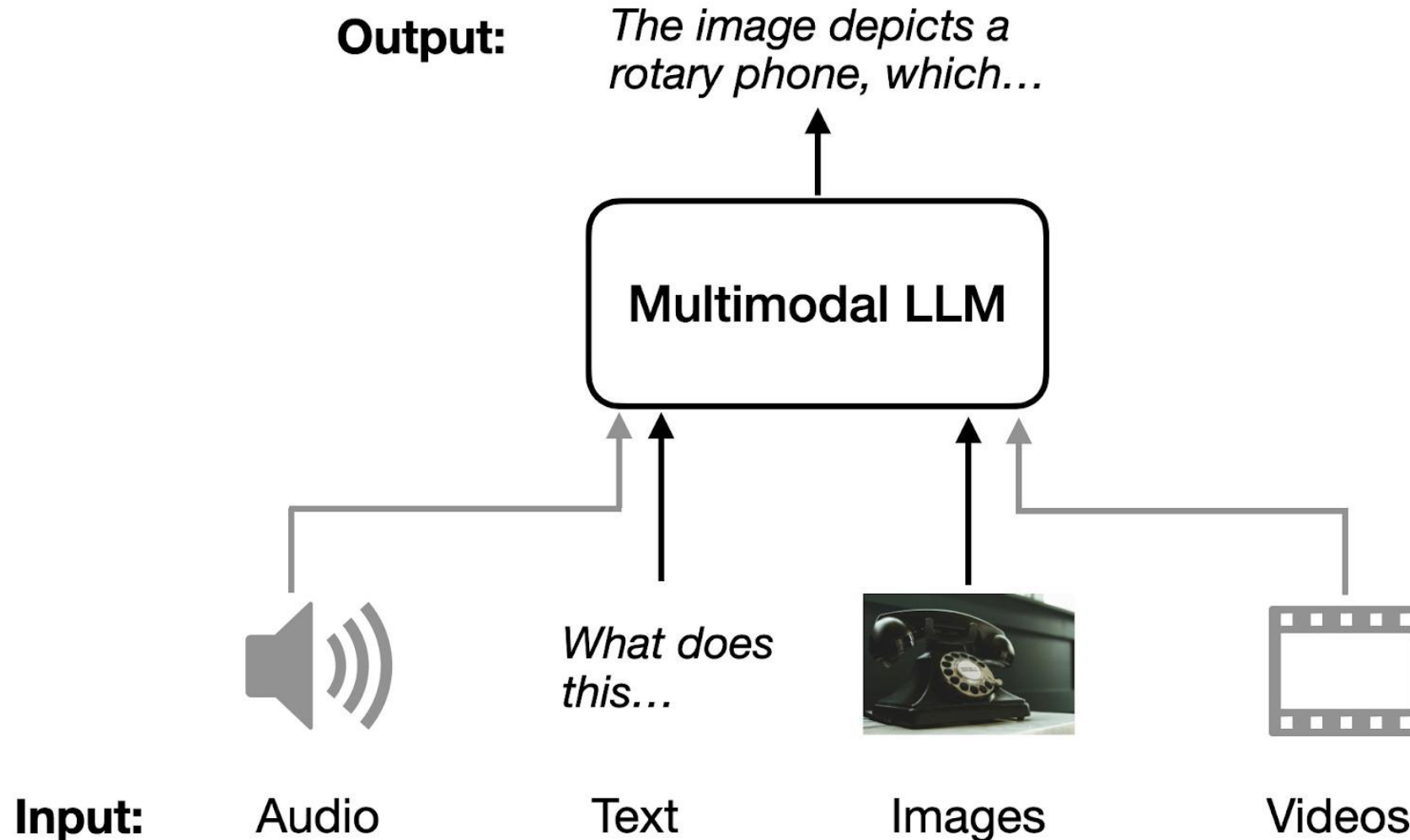
Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.



Alternatives: Agentic Scaffolding



Alternatives: MultiModal Next Token Prediction



<https://magazine.sebastianraschka.com/p/understanding-multimodal-llms>



Remarks

- ❑ We learned about GPT-X, BERT, T5 and other large pre-trained language models
- ❑ Emergent in-context learning is not yet well-understood!
- ❑ “Small” models like BERT have become general tools in a wide range of settings.
- ❑ Some tasks will just improve continually via scale and even jump (i.e., emergent behavior).
- ❑ Scaling laws are interesting for everyone!
 - Theorists (why do we get scaling laws)
 - Practitioners (lets use scaling laws to optimize)
 - AI enthusiasts (can we get AGI with more gpus?)
- ❑ Many alternatives left to explore

