CSCI 5541: Natural Language Processing

Lecture 11: Pretraining Paradigm and Scaling Law

Dongyeop Kang (DK), University of Minnesota

dongyeop@umn.edu | twitter.com/dongyeopkang | dykang.github.io

computer science & engineering

MINNESOTA NLP EST. 2021

Some slides borrowed from Anna Goldie (Google Brain)

UNIVERSITY OF MINNESOTA
Driven to Discover®

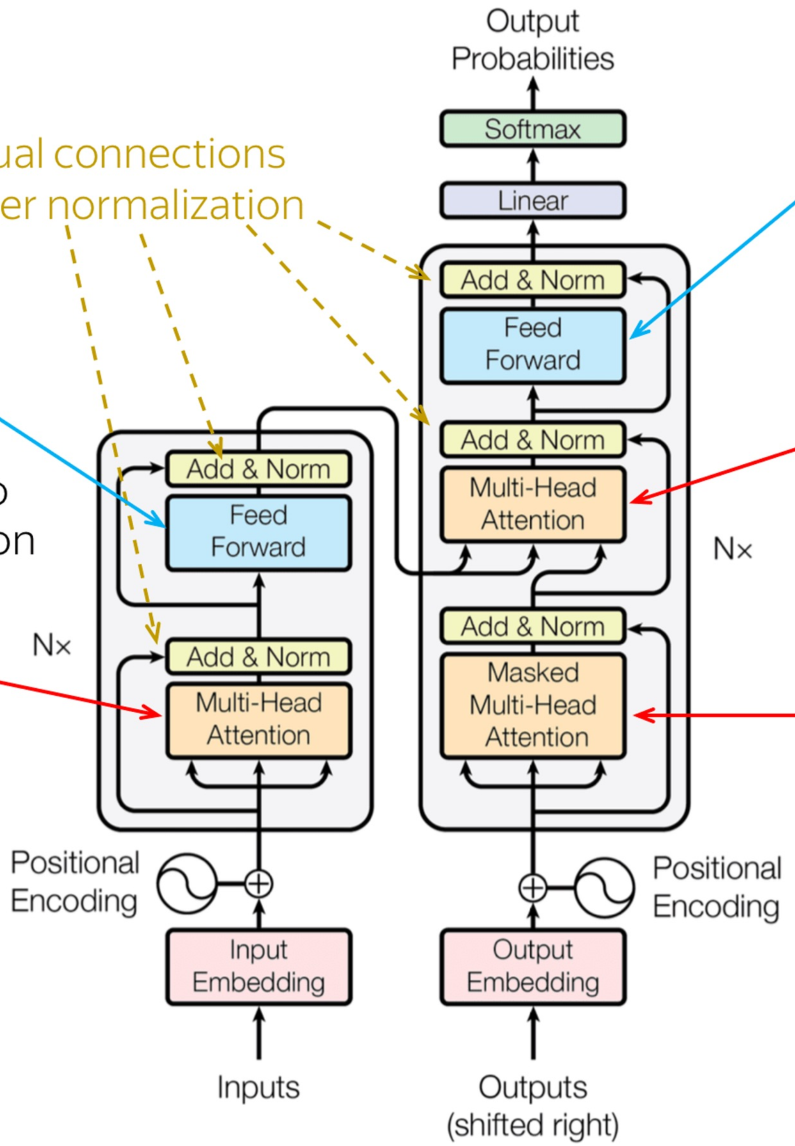
Recap

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

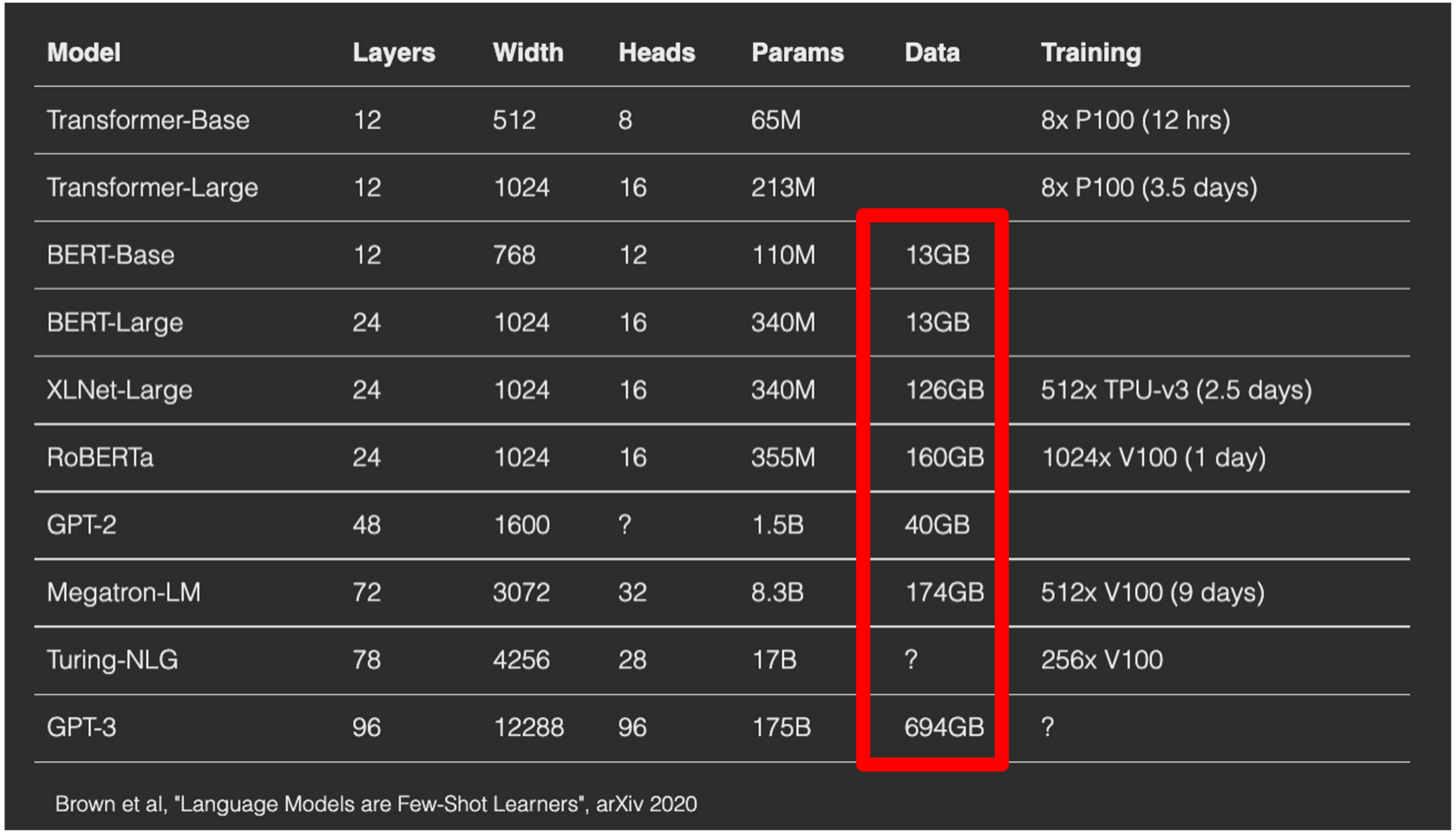
Decoder-encoder attention: target token looks at the source

queries – from decoder states; keys and values from encoder states

Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

Recap



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

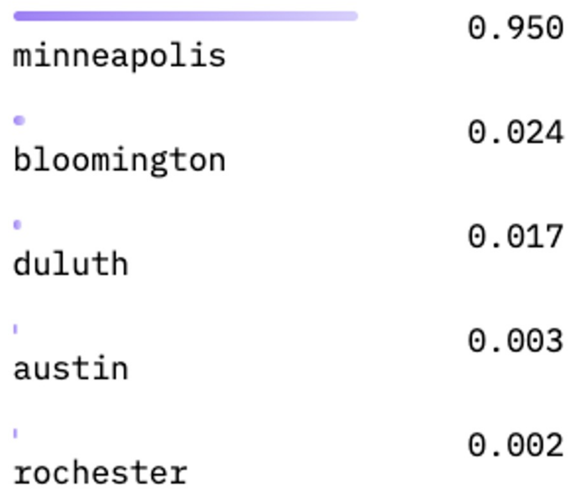
http://hal.cse.msu.edu/teaching/2020-fall-deep-learning/14-nlp-and-transformers/#/22/0/

Agenda

- ❑ What can we learn from reconstructing the input?
- ❑ Subword modeling in pretraining
- ❑ Pretraining for three types of architectures
  - ○ Encoder-only
  - ○ Decoder-only
  - ○ Encoder-Decoder
- ❑ GPT3, in-context learning, and VERY large language models
- ❑ Law of scale

What can we learn from reconstructing the input?

University of Minnesota is located in _ _ _ _ _, Minnesota.

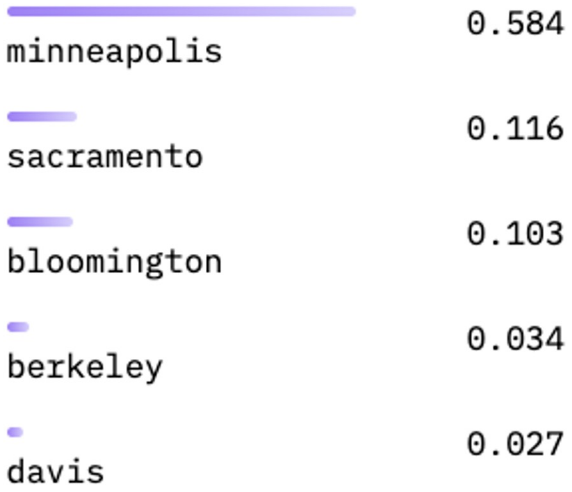


minneapolis	0.950
bloomington	0.024
duluth	0.017
austin	0.003
rochester	0.002

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

What can we learn from reconstructing the input?

University of Minnesota is located in _ _ _ _ _, California.

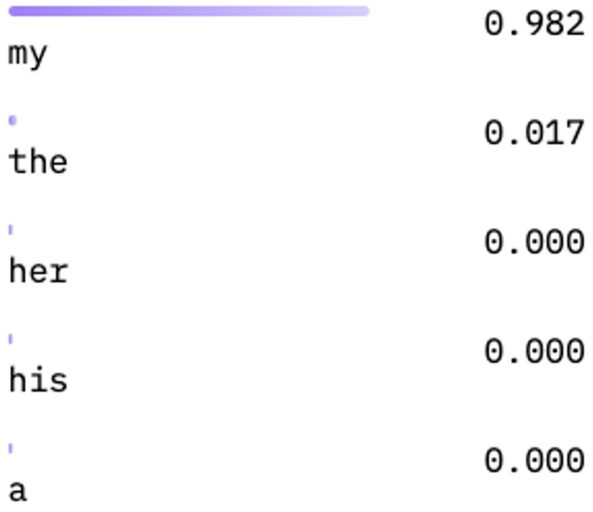


minneapolis	0.584
sacramento	0.116
bloomington	0.103
berkeley	0.034
davis	0.027

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

What can we learn from reconstructing the input?

I put _ _ _ fork down on the table.

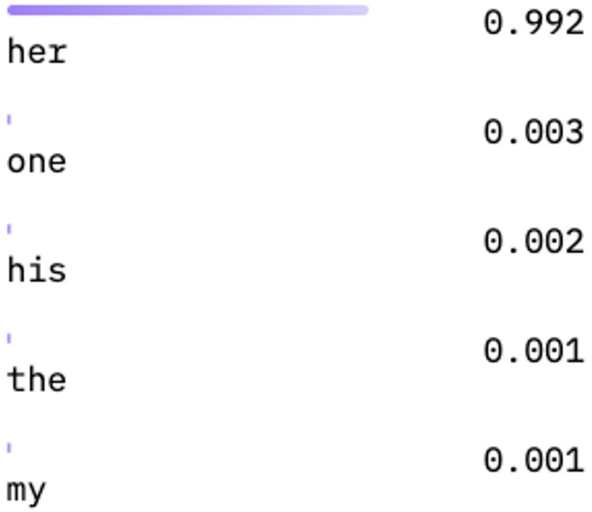


my	0.982
the	0.017
her	0.000
his	0.000
a	0.000

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

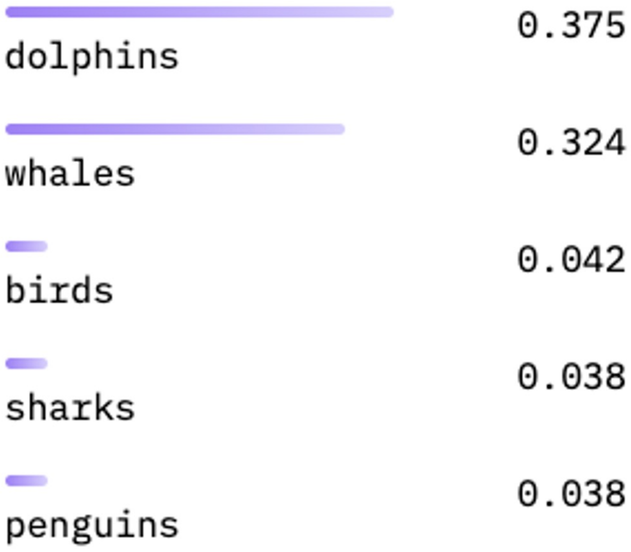


her	0.992
one	0.003
his	0.002
the	0.001
my	0.001

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 _ _ _ _ _.

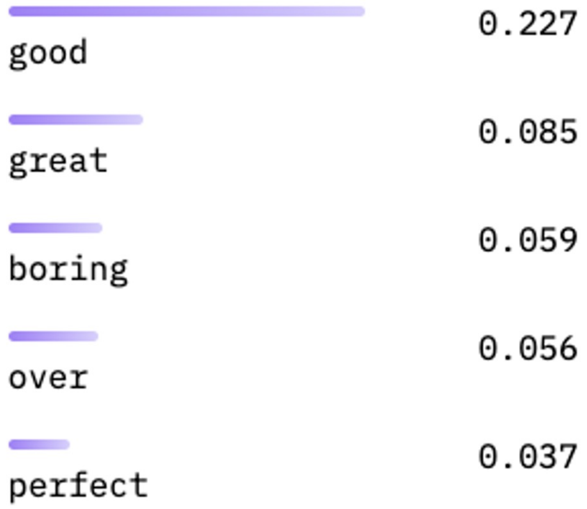


dolphins	0.375
whales	0.324
birds	0.042
sharks	0.038
penguins	0.038

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 ___.

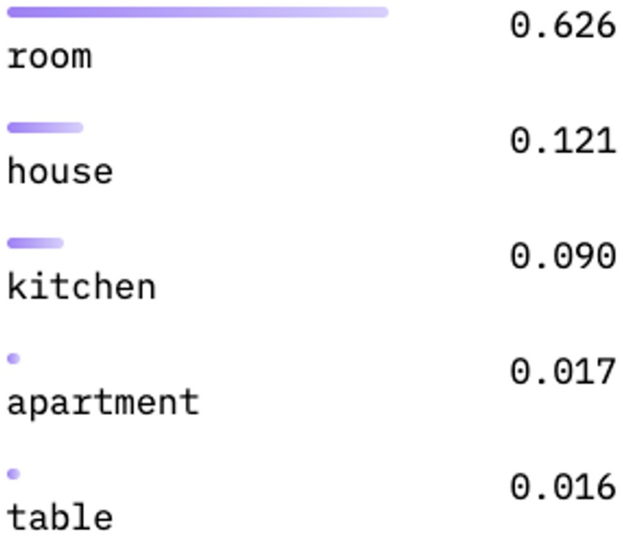


good	0.227
great	0.085
boring	0.059
over	0.056
perfect	0.037

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 ______

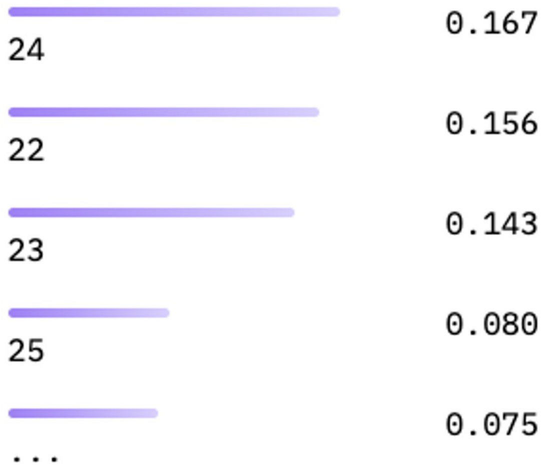


room	0.626
house	0.121
kitchen	0.090
apartment	0.017
table	0.016

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, ____

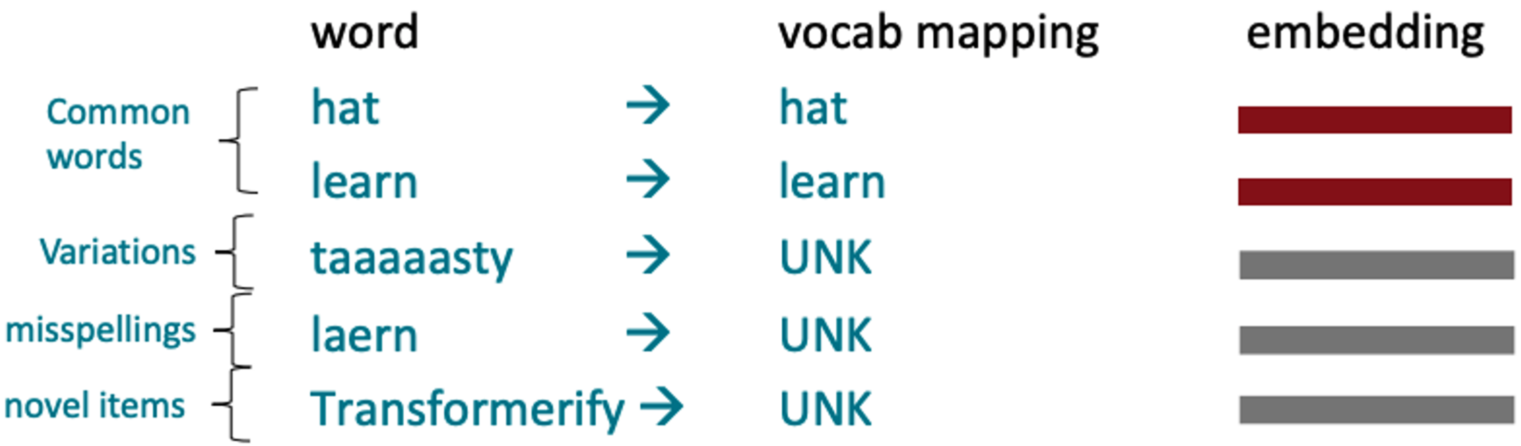


24	0.167
22	0.156
23	0.143
25	0.080
...	0.075

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

Brief notes on subword modeling

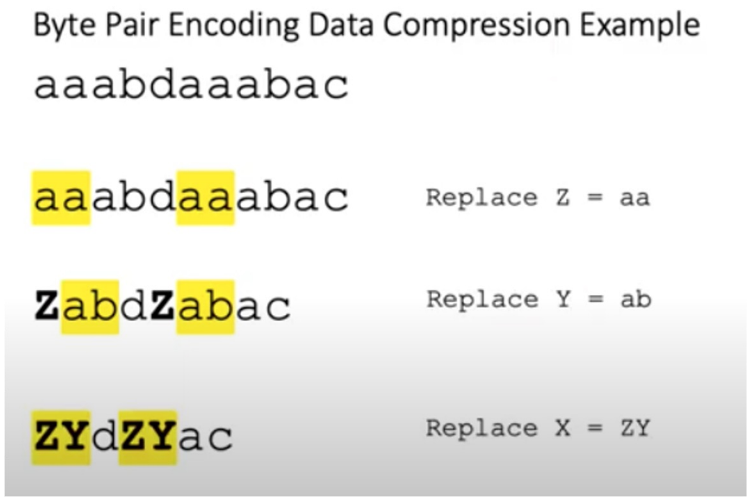
- ❑ We assume a fixed vocab of tens of thousands of words, built from train set.
- ❑ All novel words seen at test time are mapped to a single UNK token.
- ❑ Finite vocabulary assumptions make even less sense in many languages.
    - ○ Many languages exhibit complex morphology, or word structure.
    - ○ *Swahili* verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)



	word		vocab mapping	embedding
Common words	hat	→	hat	🟥
	learn	→	learn	🟥
Variations	taaaaasty	→	UNK	▬
misspellings	laern	→	UNK	▬
novel items	Transformerify	→	UNK	▬

The byte-pair encoding algorithm

- Subword modeling in NLP encompasses a wide range of methods for reasoning about structure below the word level. (Parts of words, characters, bytes.)
    - The dominant modern paradigm is to learn a vocabulary of parts of words (subword tokens).
- Byte-pair encoding is a simple, effective strategy for subword modeling
    - 1. Start with a vocabulary containing only characters and an "end-of-word" symbol.
    - 2. Using a corpus of text, find the most common pair of adjacent characters "a,b"; add subword "ab" to the vocab.
    - 3. Replace instances of the character pair with the new subword; repeat until desired vocab size.
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.



Byte Pair Encoding Data Compression Example

```
aaabdaaabac

aaabdaaabac     Replace Z = aa

ZabdZabac       Replace Y = ab

ZYdZYac         Replace X = ZY
```

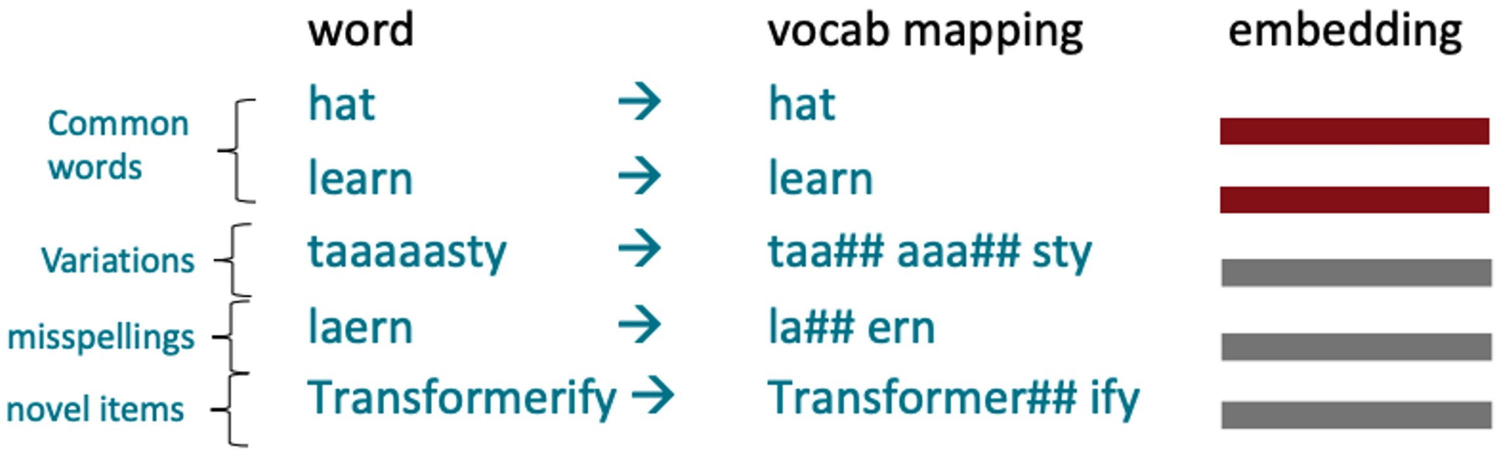
https://en.wikipedia.org/wiki/Byte_pair_encoding

Neural Machine Translation of Rare Words with Subword Units, ACL 2016

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016

Word structure and subword models

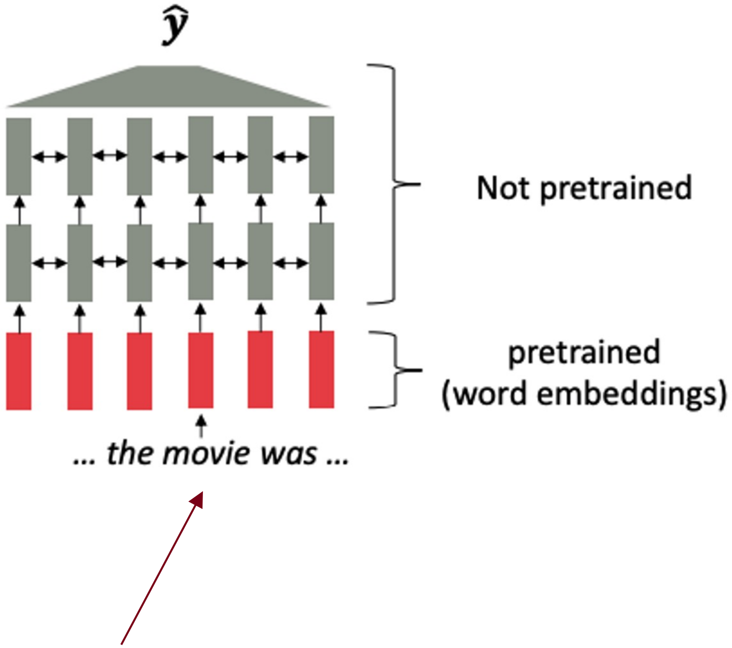
- ❑ Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.
  - ○ In the worst case, words are split into as many subwords as they have characters.



	word		vocab mapping	embedding
Common words	hat	→	hat	🟥
	learn	→	learn	🟥
Variations	taaaaasty	→	taa## aaa## sty	▬
misspellings	laern	→	la## ern	▬
novel items	Transformerify	→	Transformer## ify	▬

Recap: pre-trained word embeddings

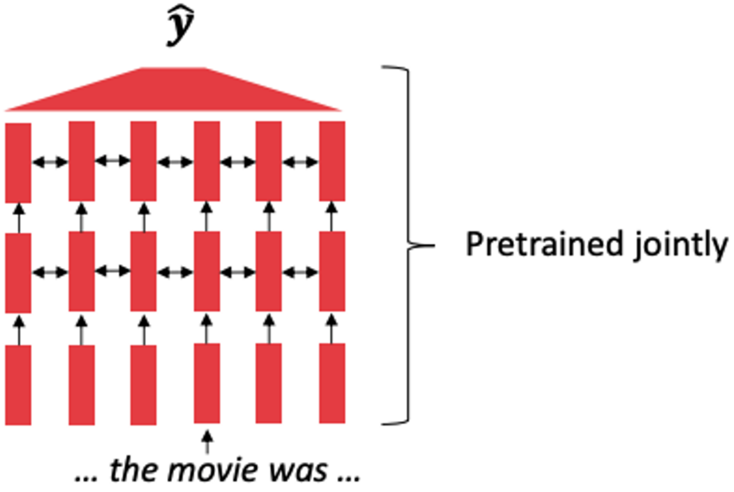
- Before 2017:
  - Start with pretrained word embeddings (no context!)
  - Learn how to incorporate context in an LSTM or Transformer while training on the task.
- Some issues to think about:
  - The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.
  - Most of the parameters in our network are randomly initialized!



movie gets the same word embeddings, regardless of what sentence it appears.

Recap: pre-trained **whole** embeddings

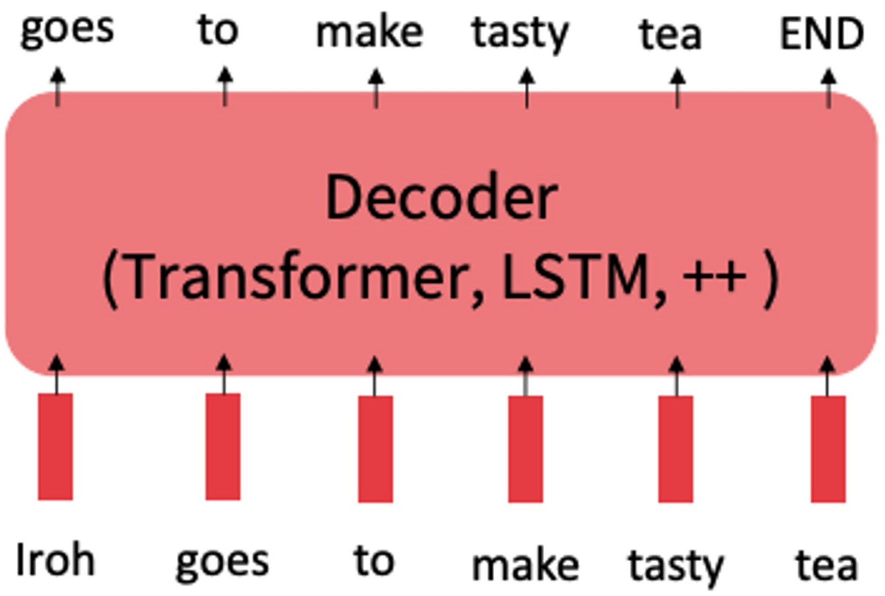
- In modern NLP:
  - All (or almost all) parameters in NLP networks are initialized via pretraining.
  - Pretraining methods hide parts of the input from the model, then train the model to reconstruct those parts
- This has been exceptionally effective at building strong:
  - representations of language
  - parameter initializations for strong NLP models.
  - probability distributions over language that we can sample from



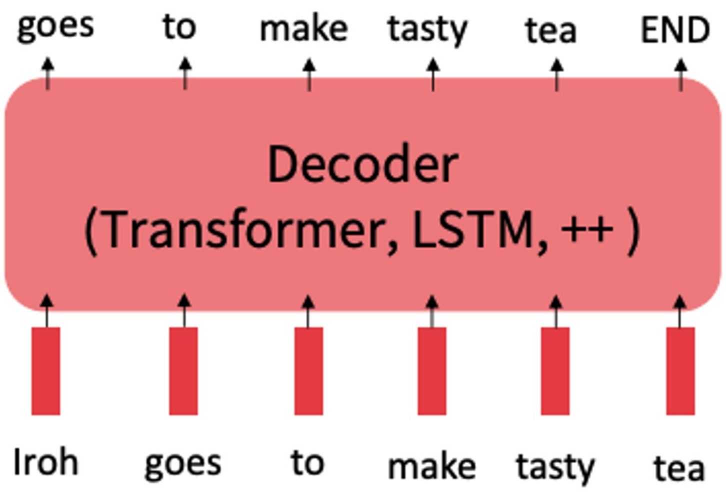
This model has learned how to represent entire sentences through pretraining

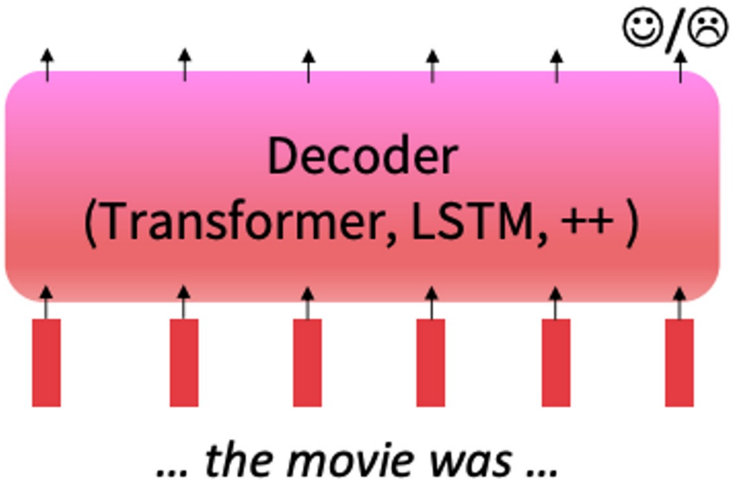
Pretraining through language modeling

- 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

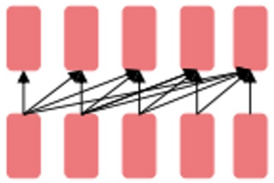




Step 1: Pretrain (on language modeling)
Lots of text; learn general things!
Serve as parameter initialization.

Step 2: Finetune (on your task)
Not many labels; adapt to the task!

Pretraining for three types of architectures

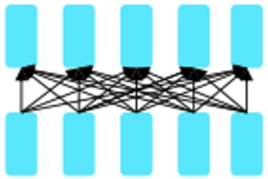


Decoders

❑ Simple left-to-right language models!

❑ Nice to generate from; can’t condition on future words

❑ Examples: GPT-2, GPT-3, LaMDA

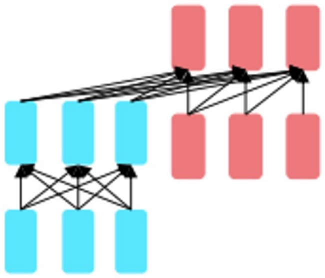


Encoders

❑ 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

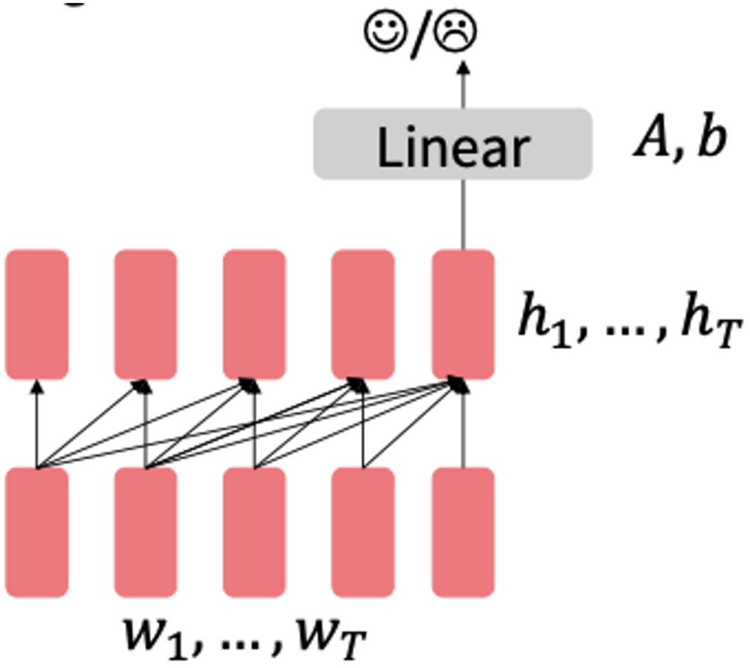
Pretraining and finetuning decoders

- ❑ 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.

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

where A and b are randomly initialized and specified by the downstream task.

- ❑ Gradients backpropagate through the whole network.



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

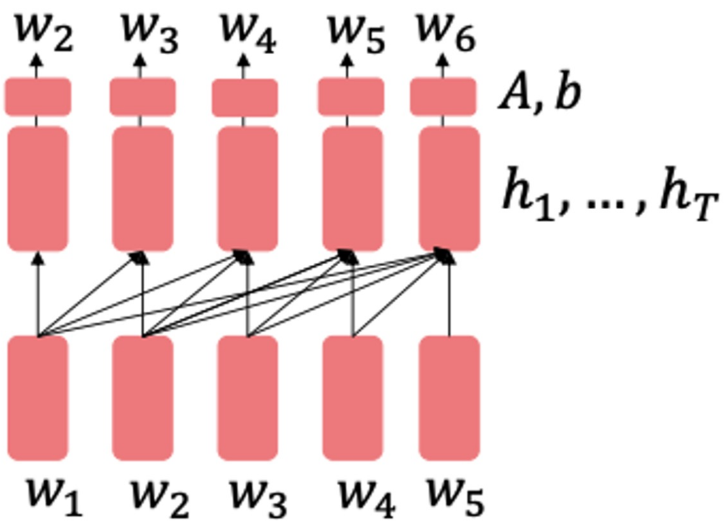
Pretraining and finetuning decoders

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

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

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.]

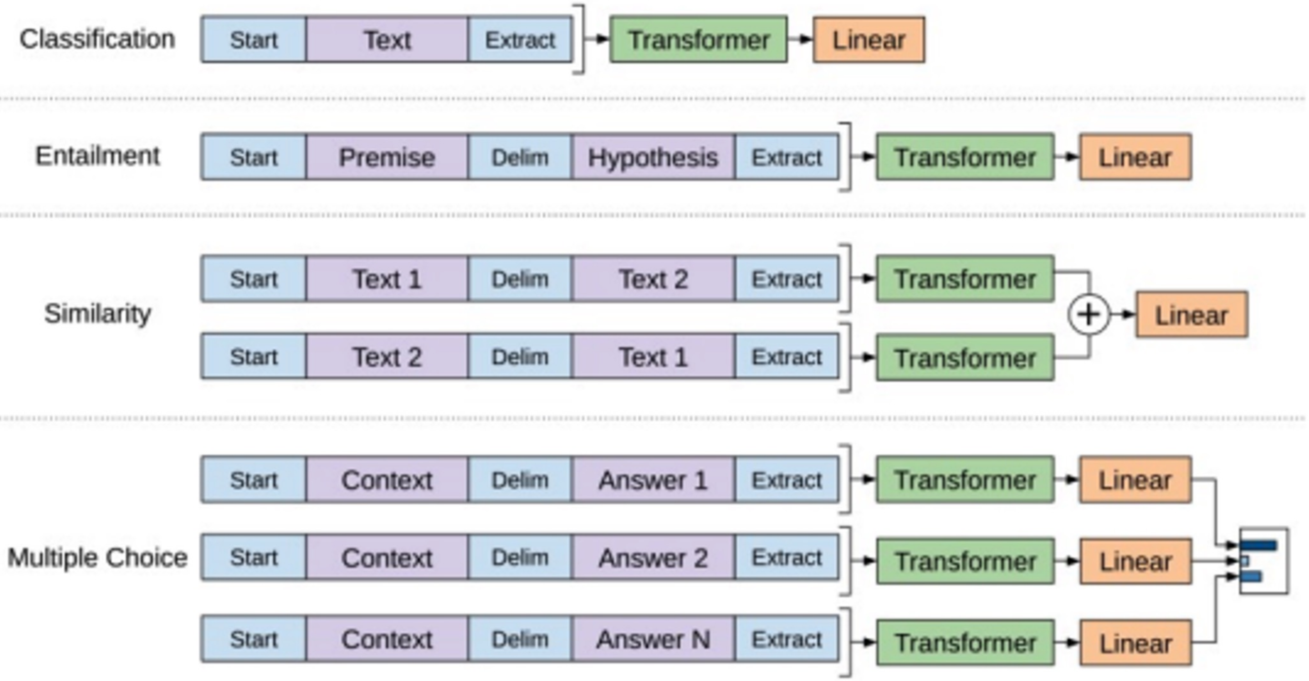
Generative Pretrained Transformer (GPT) (Radford et al., 2018)

- 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.

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

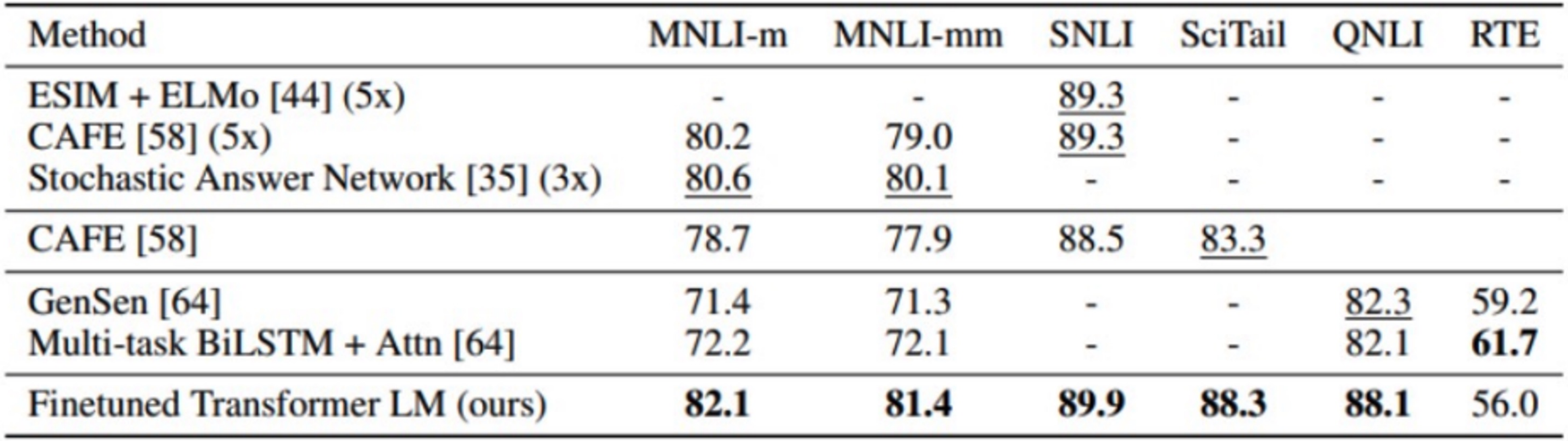
- How do we format inputs to our decoder for finetuning tasks?



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

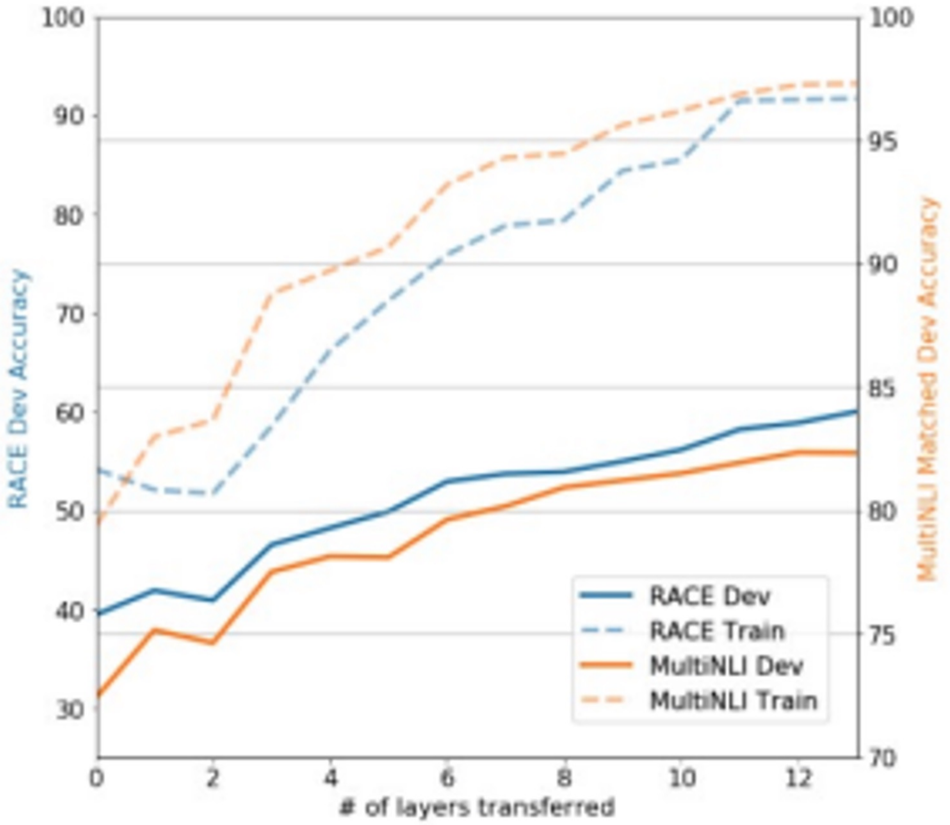
Generative Pretrained Transformer (GPT) (Radford et al., 2018)

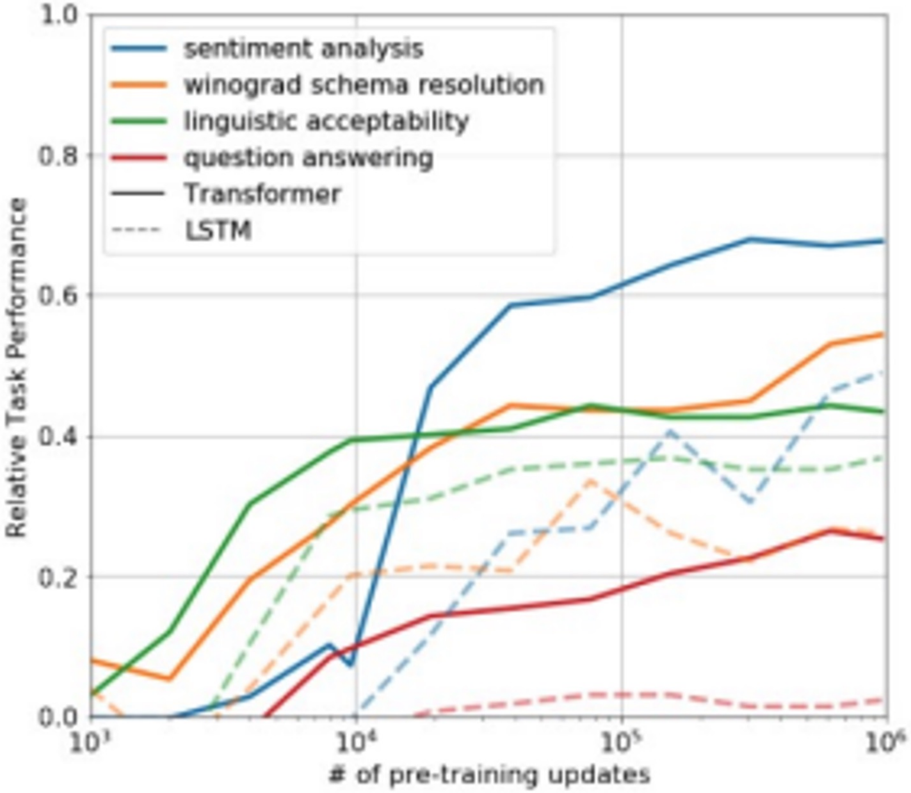
❑ GPT results on various natural language inference datasets.



Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	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

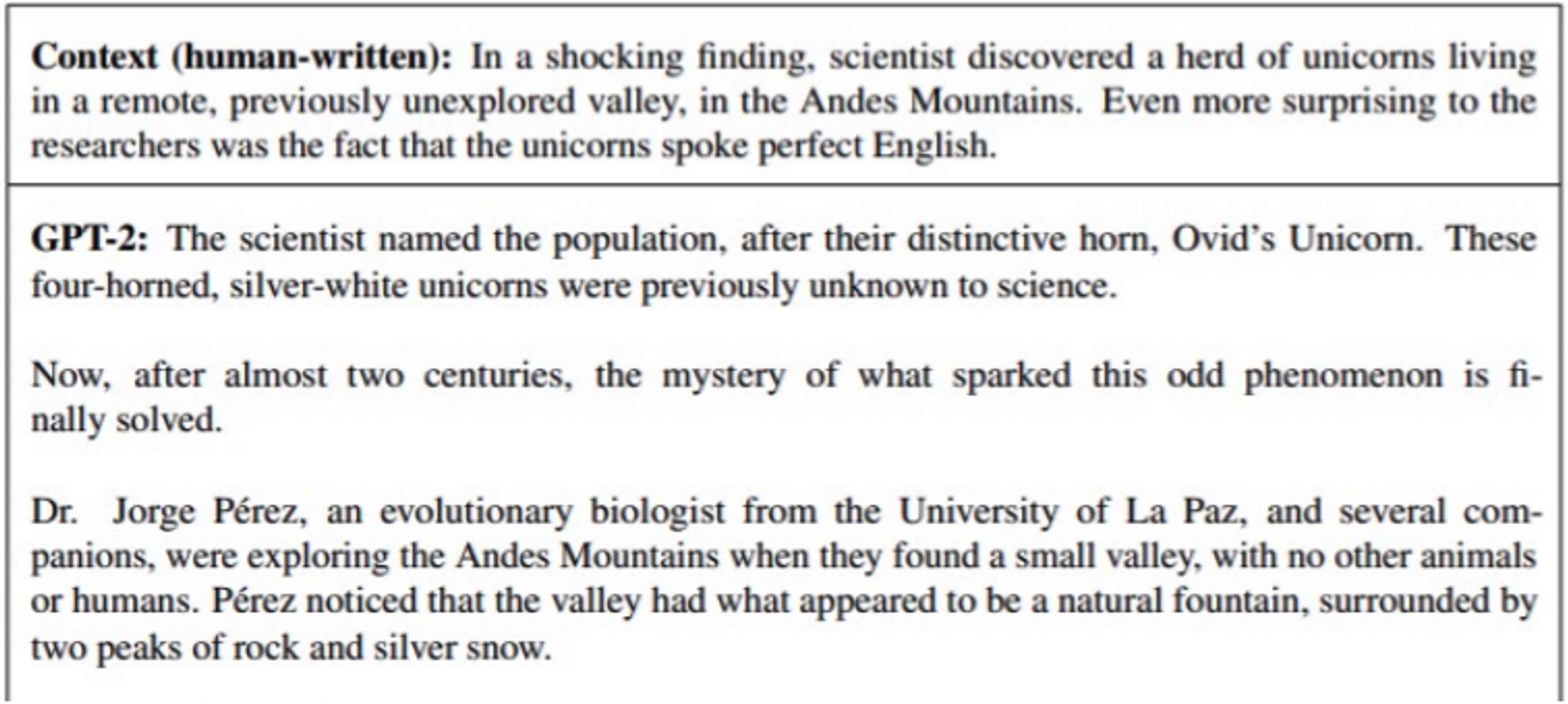
Effect of Pretraining in GPT





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



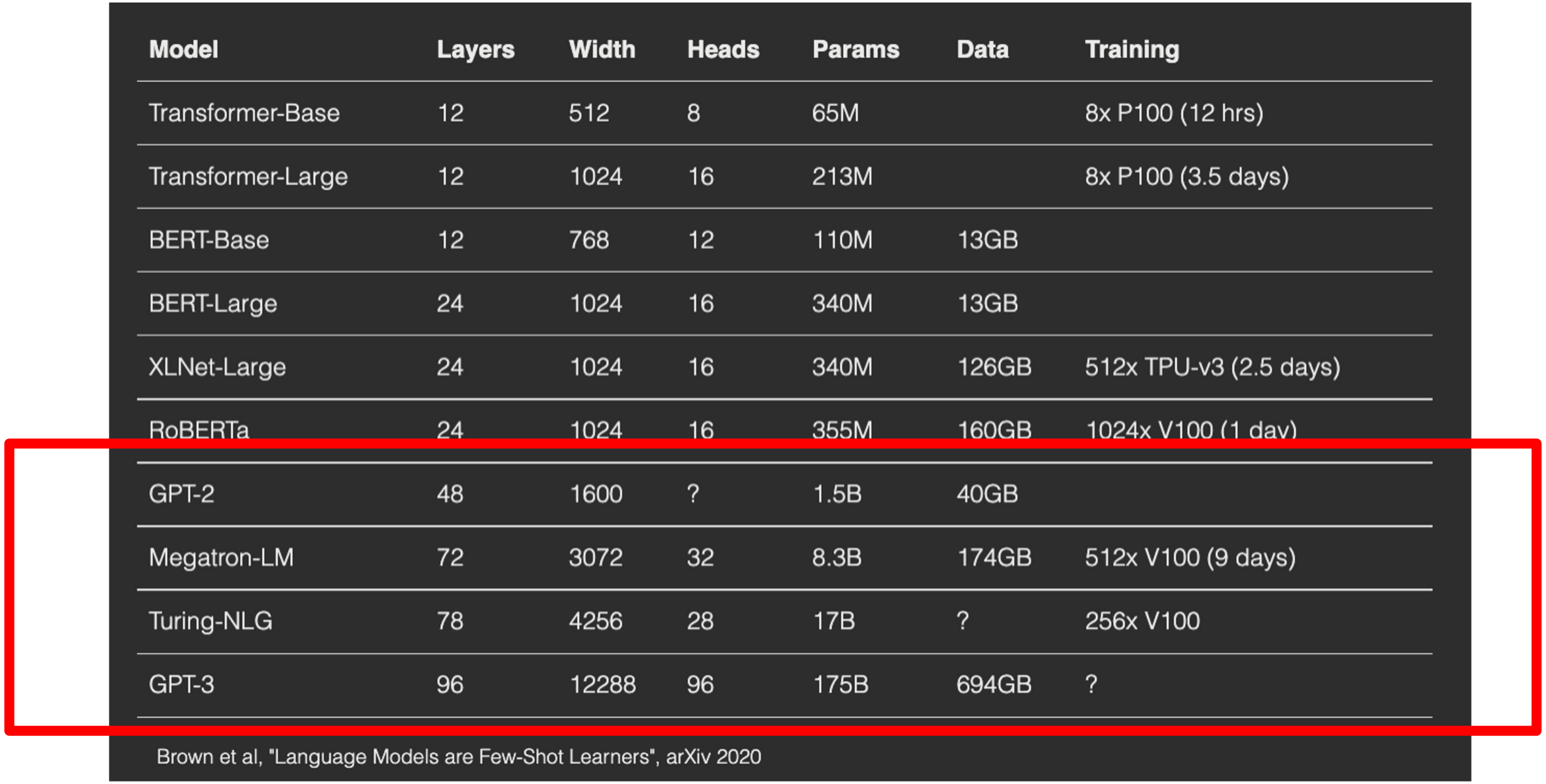
Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

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.

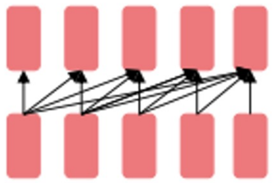
Generative Pretrained Transformer (GPT)



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

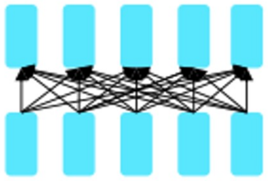


Decoders

❑ Simple left-to-right language models!

❑ Nice to generate from; can’t condition on future words

❑ Examples: GPT-2, GPT-3, LaMDA

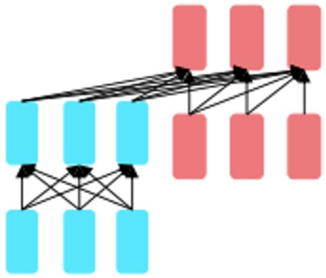


Encoders

❑ Gets bidirectional context – can condition on future!

❑ Masked language models

❑ Examples: BERT, RoBERTa



Encoder-Decoders

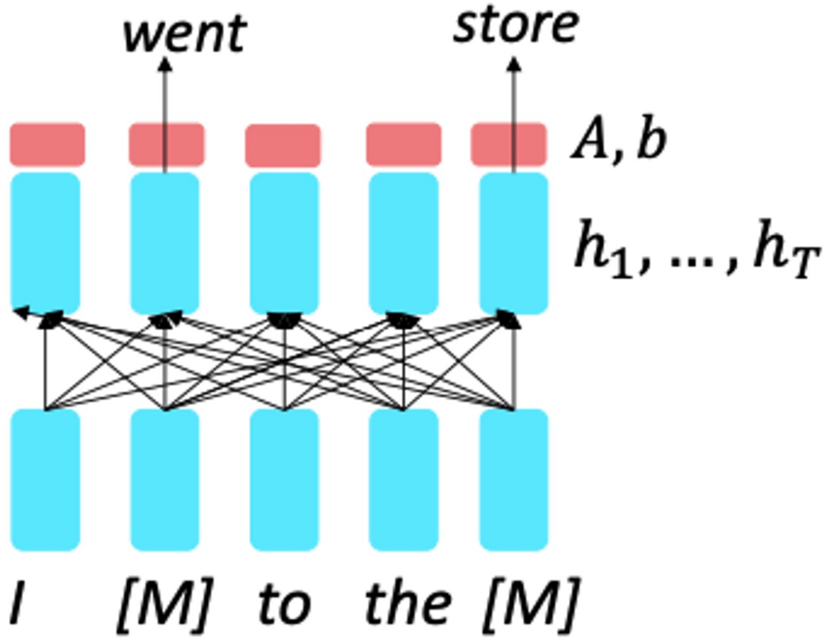
❑ Good parts of decoders and encoders?

❑ What’s the best way to pretrain them?

❑ Examples: T5, BART

Pretraining and finetuning encoders

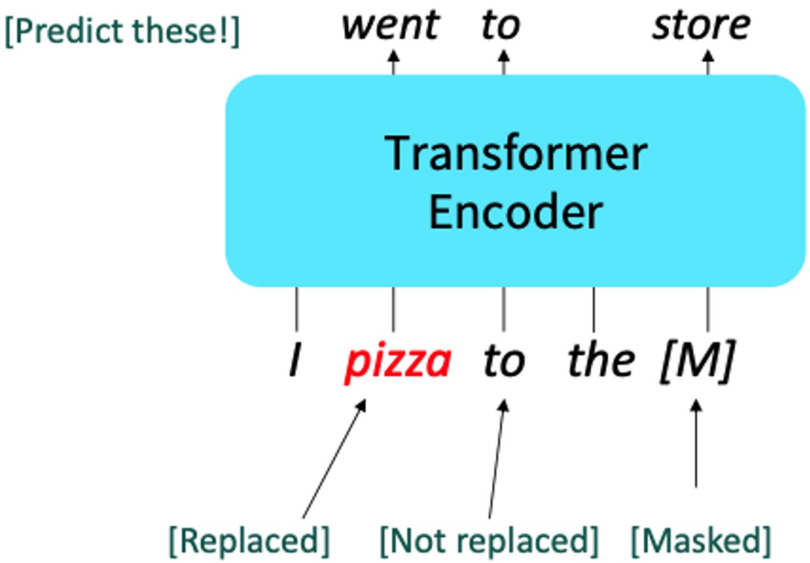
- So far, we've looked at language model pretraining. But, encoders get bidirectional context, so we can't do language modeling!
- Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.
- Only add loss terms from words that are "masked out." If $\hat{x}$ is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



BERT: Bidirectional Encoder Representations from Transformers

(Devlin et al., 2018)

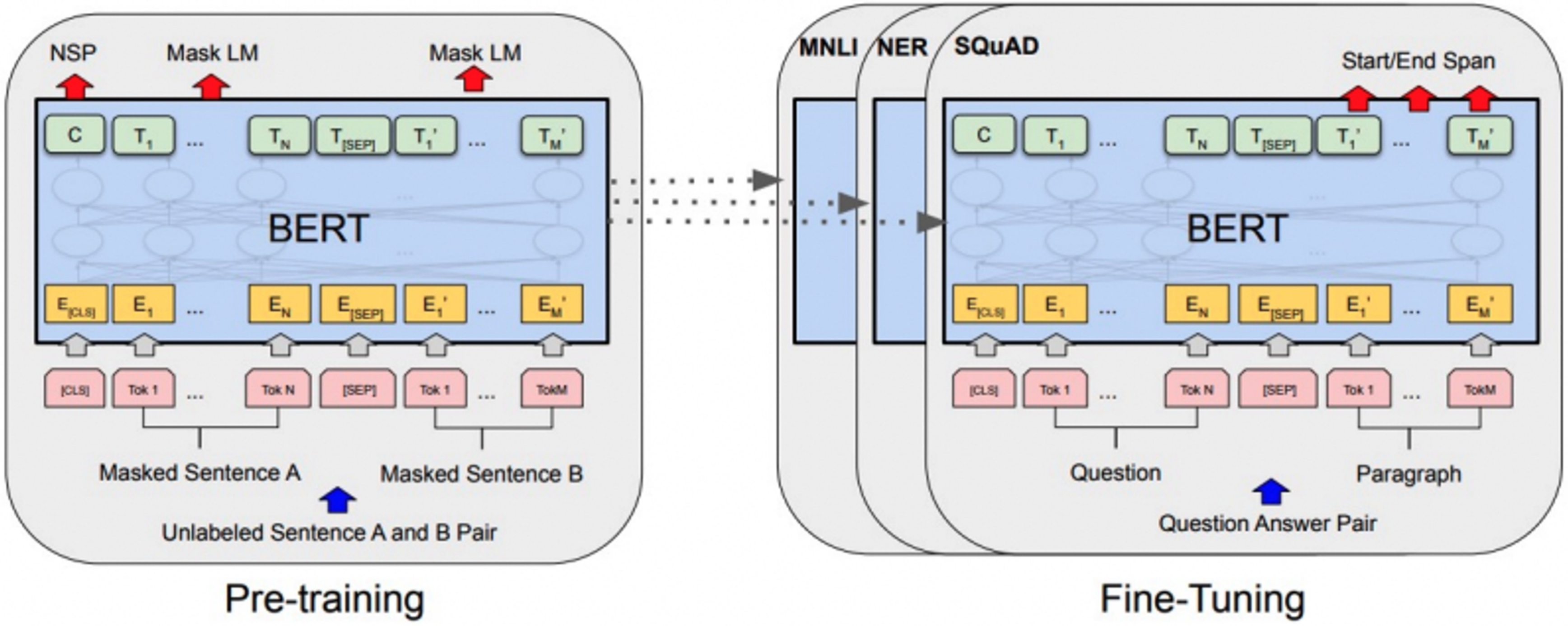
- ❑ Devlin et al., 2018 proposed the "Masked LM" objective and released the weights of their pretrained Transformer (BERT).
- ❑ Details about Masked LM for BERT:
  - ○ Predict a random 15% of (sub)word tokens.
  - ○ Replace input word with [MASK] 80% of the time
  - ○ Replace input word with a random token 10% of the t
  - ○ Leave input word unchanged 10% of the time (but stil 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!)



BERT: Bidirectional Encoder Representations from Transformers

(Devlin et al., 2018)

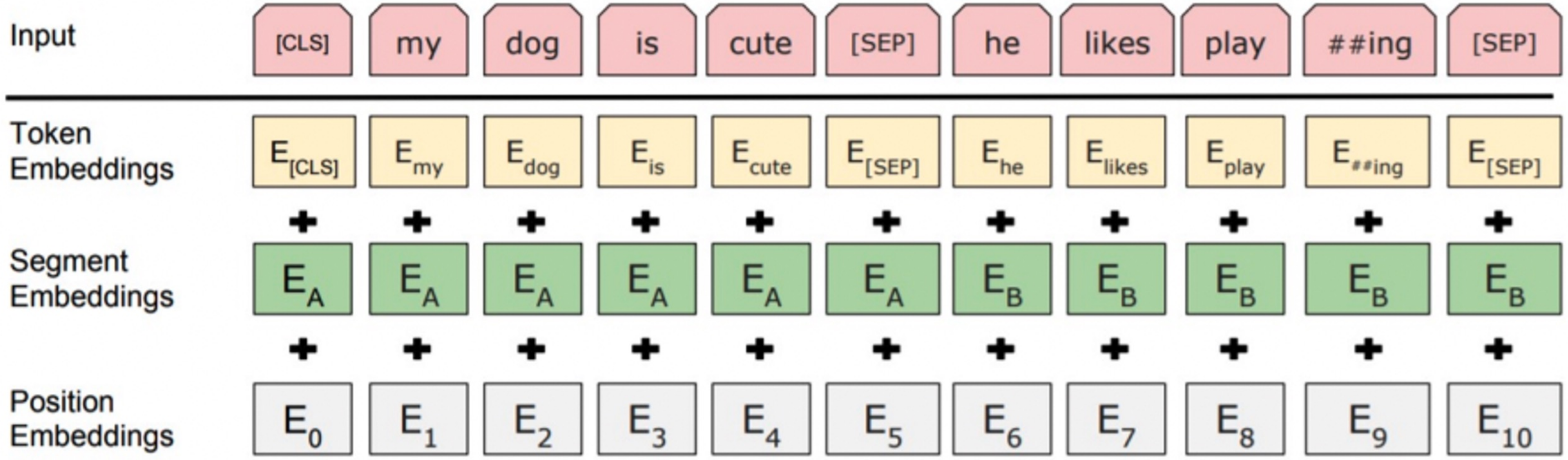
- ❑ Unified Architecture: As shown below, there are minimal differences between the pre-training architecture and the fine-tuned version for each downstream task



BERT: Bidirectional Encoder Representations from Transformers

(Devlin et al., 2018)

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



Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

- 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.

(Devlin et al., 2018)

Details about BERT Training

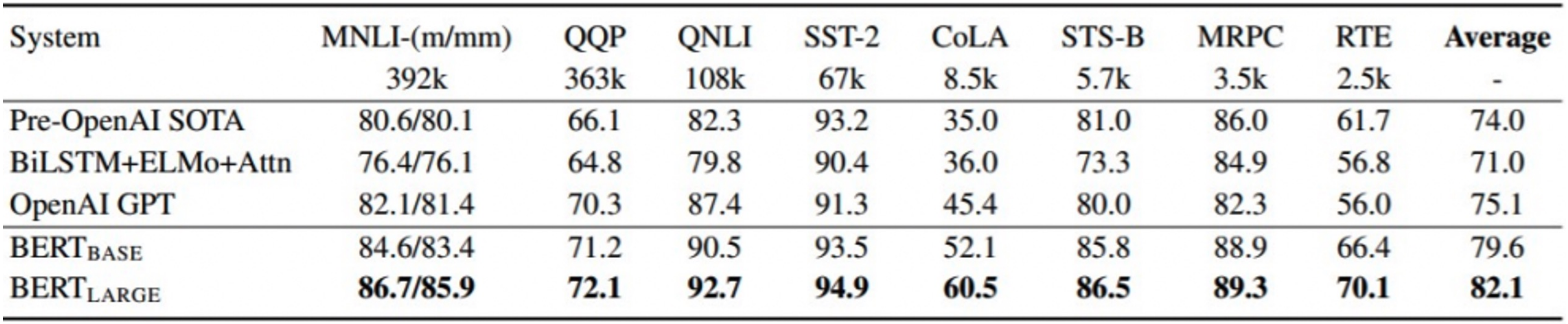
- ❑ Two models were released:
  - BERT-base: 12 layers, 768-dim hidden, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden, 16 attention heads, 340 million params.
- ❑ Trained on:
  - BookCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- ❑ Pretraining 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

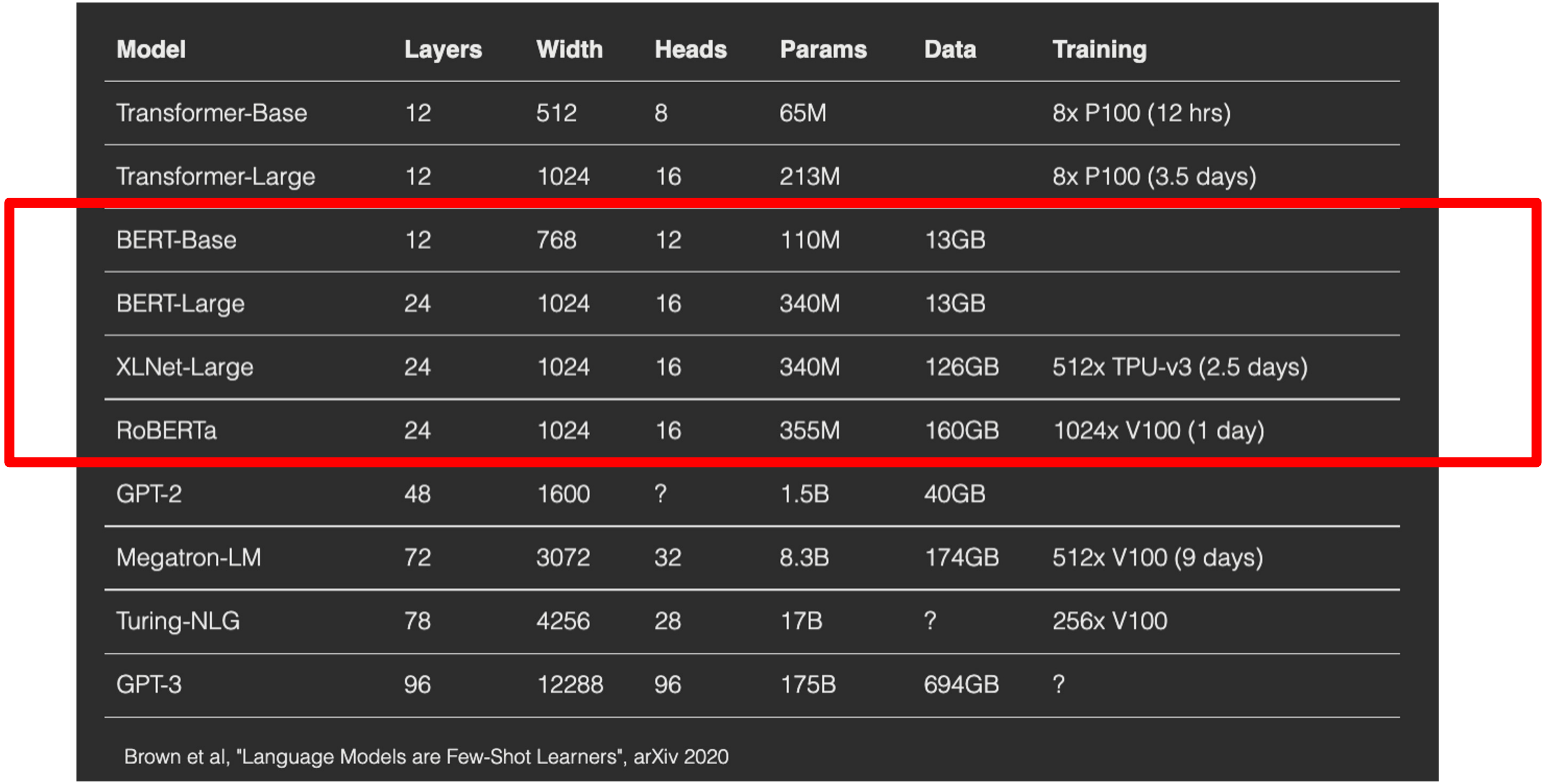


System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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



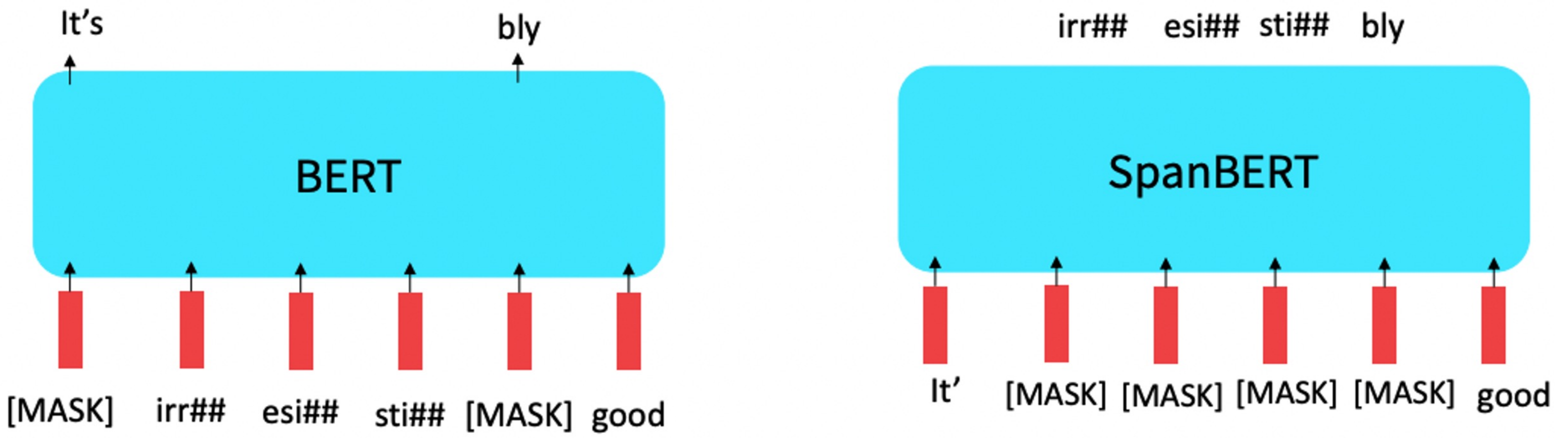
Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hrs)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13GB	
BERT-Large	24	1024	16	340M	13GB	
XLNet-Large	24	1024	16	340M	126GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160GB	1024x 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

(Liu et al., 2019; Joshi et al., 2020)

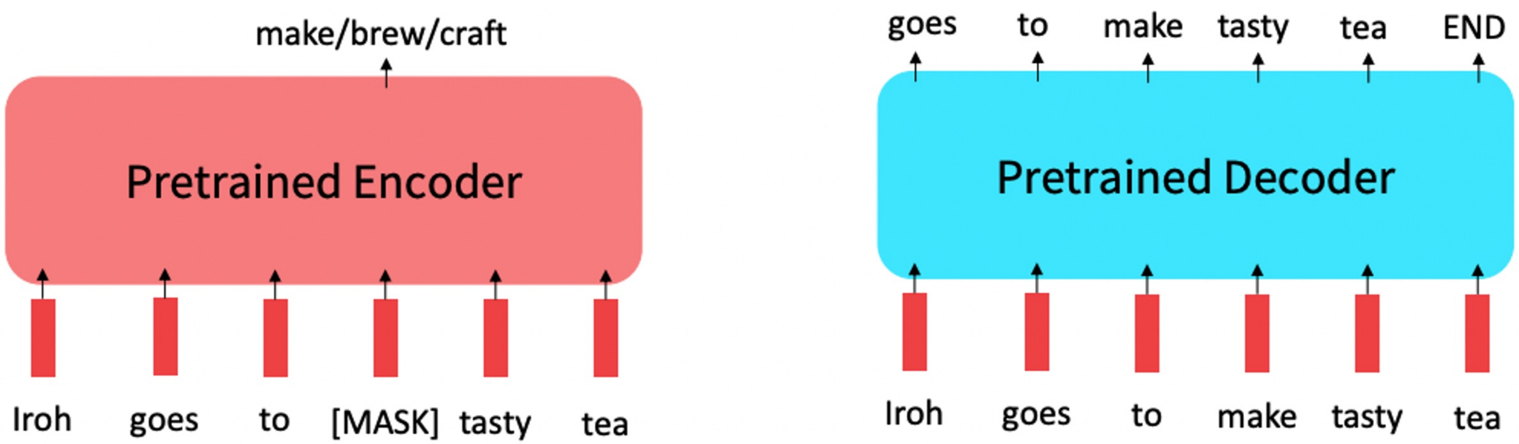
Extension of BERT

- 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

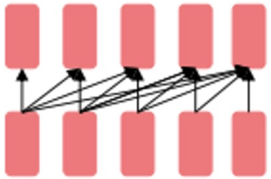


Limitations of pretrained encoders

- 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.

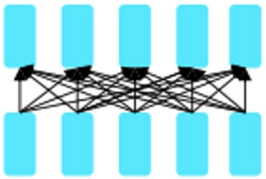


Pretraining for three types of architectures



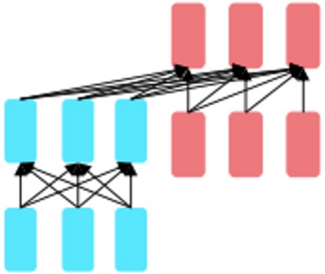
Decoders

- 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



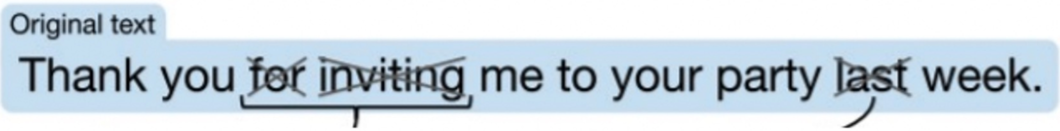
Encoder-Decoders

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

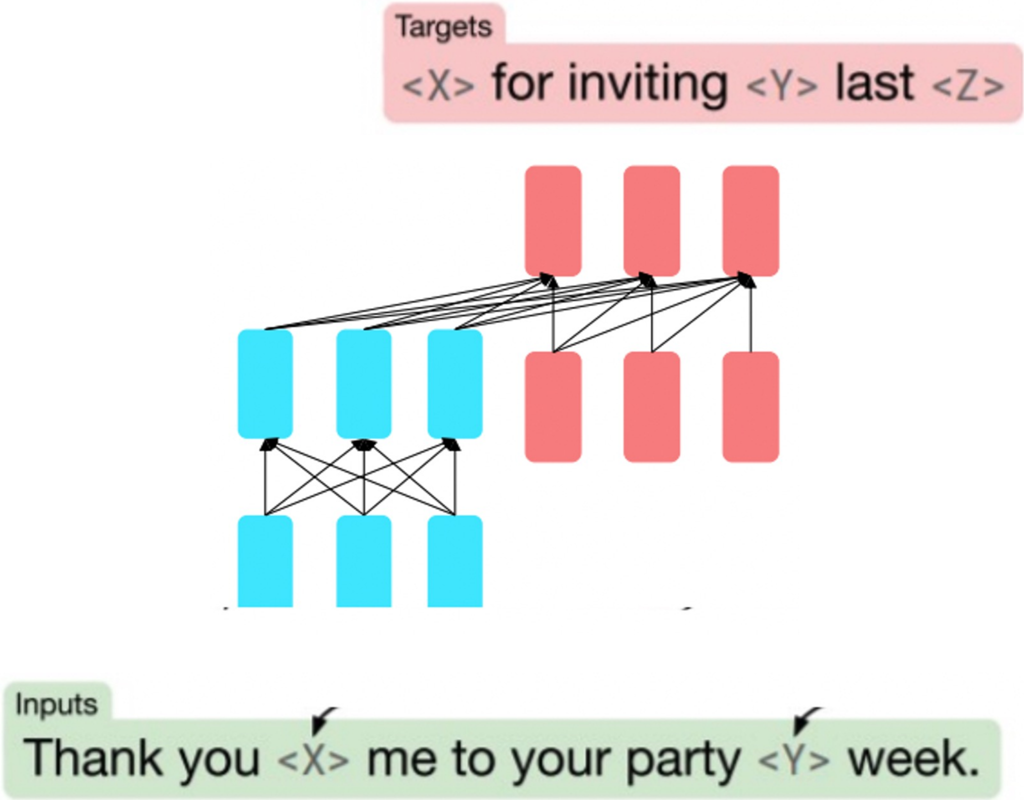
Pretraining encoder-decoders

- ❑ 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 (<x>, <y>); decode out the spans that were removed!



Thank you ~~for inviting~~ me to your party ~~last~~ week.

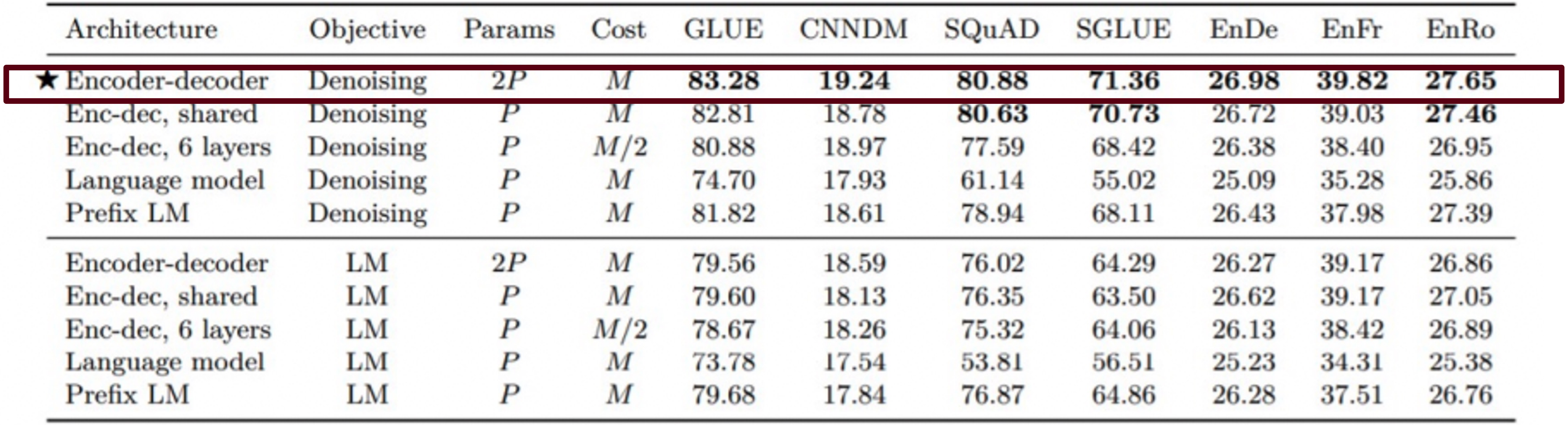


<X> for inviting <Y> last <Z>

Thank you <X> me to your party <Y> 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.



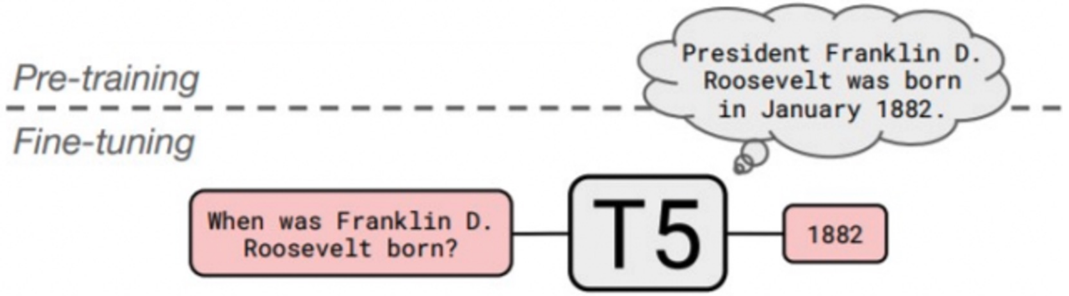
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	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

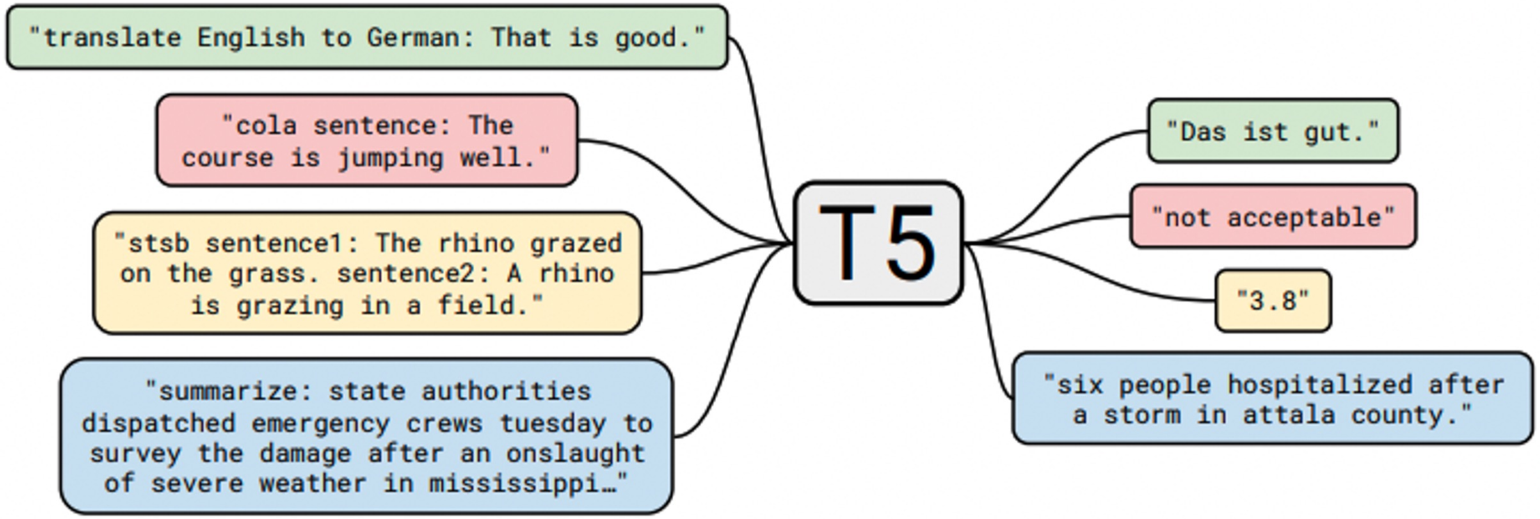
(Raffel et al., 2018)

Pretraining encoder-decoders

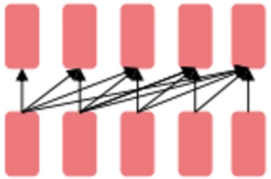
A fascinating property of T5:

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


Pre-training
Fine-tuning
President Franklin D. Roosevelt was born in January 1882.
When was Franklin D. Roosevelt born?
T5
1882

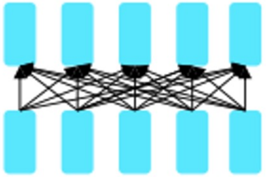

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Pretraining for three types of architectures



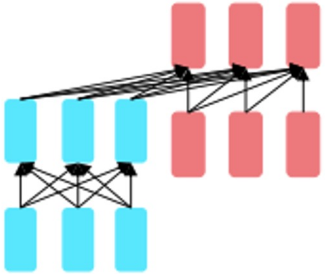
Decoders

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



Encoders

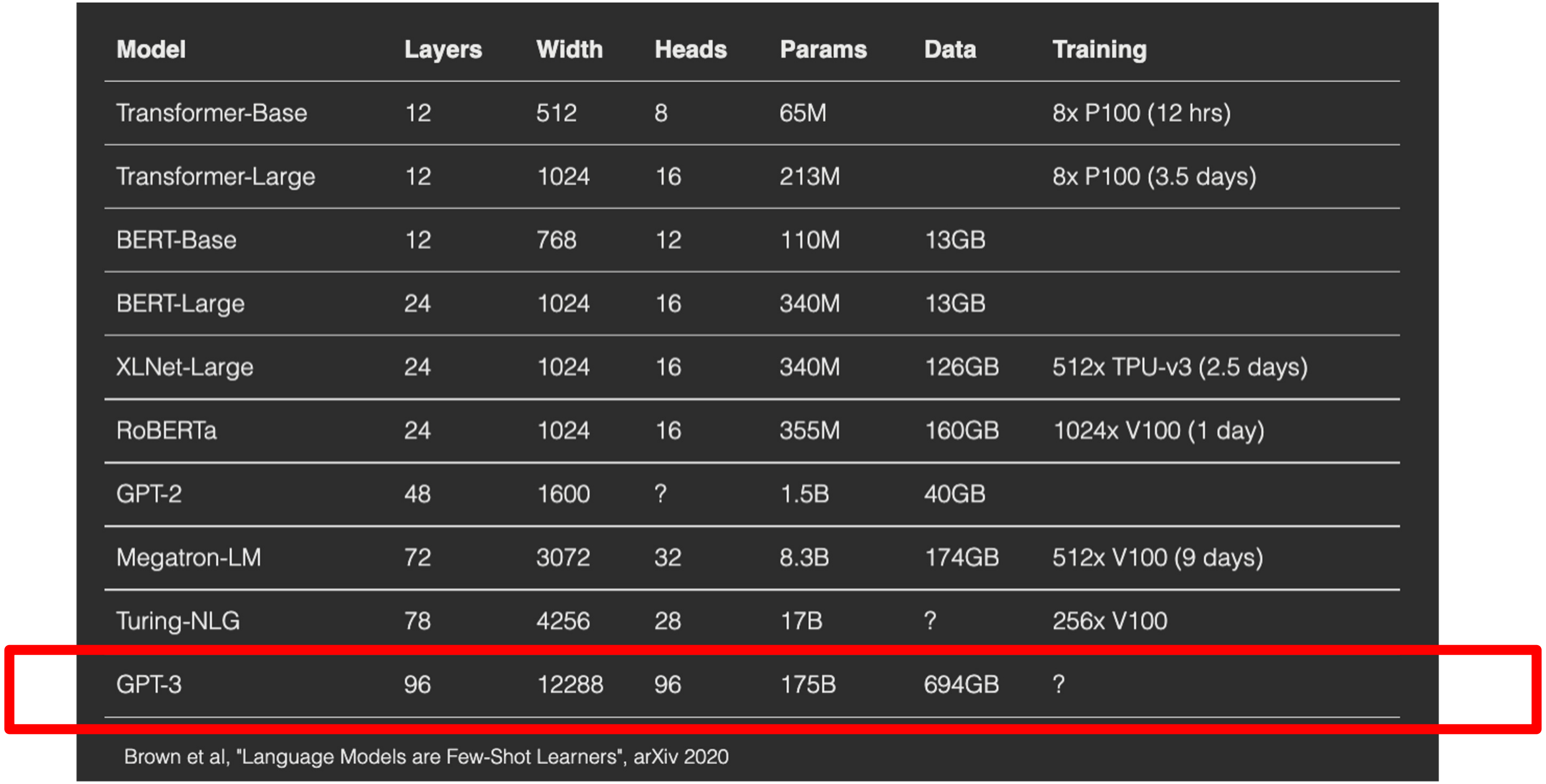
- Gets bidirectional context – can condition on future!
- Masked language models
- Examples: BERT, RoBERTa



Encoder-Decoders

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

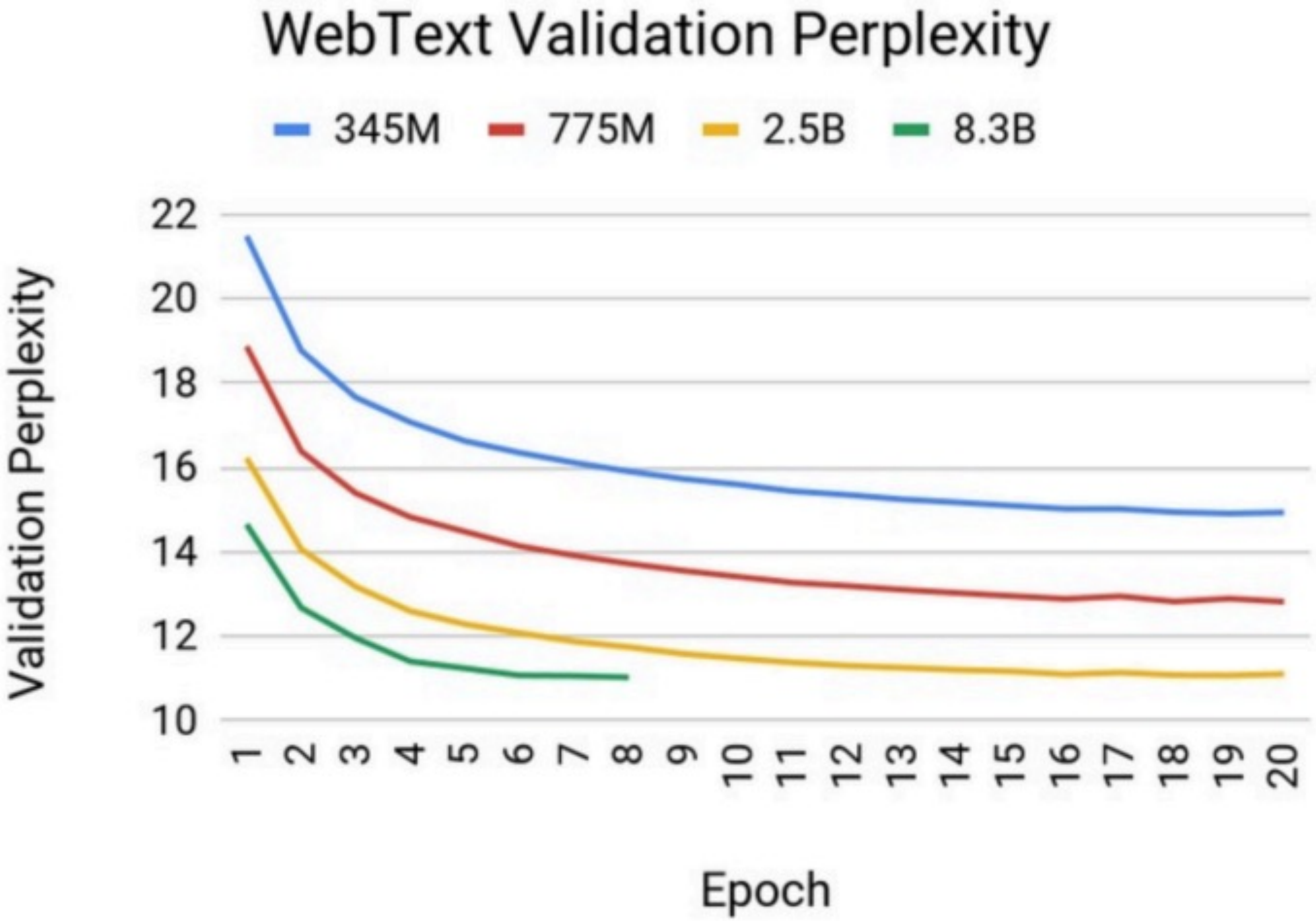
GPT3, in-context learning, and VERY large language models



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	?

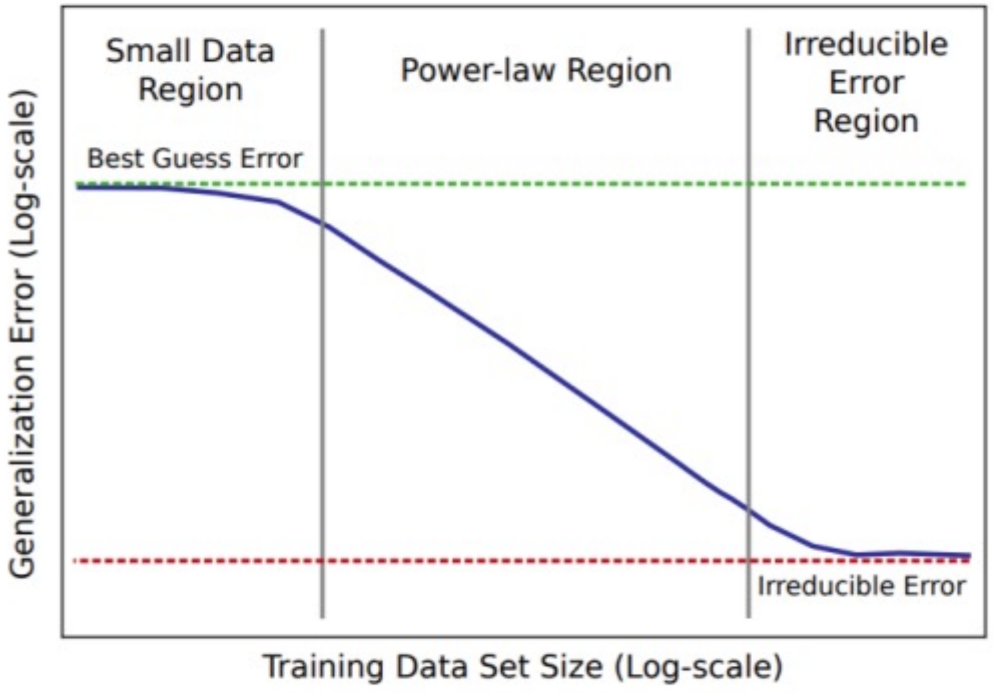
Brown et al, "Language Models are Few-Shot Learners", arXiv 2020

What are the scaling limits of large language models?

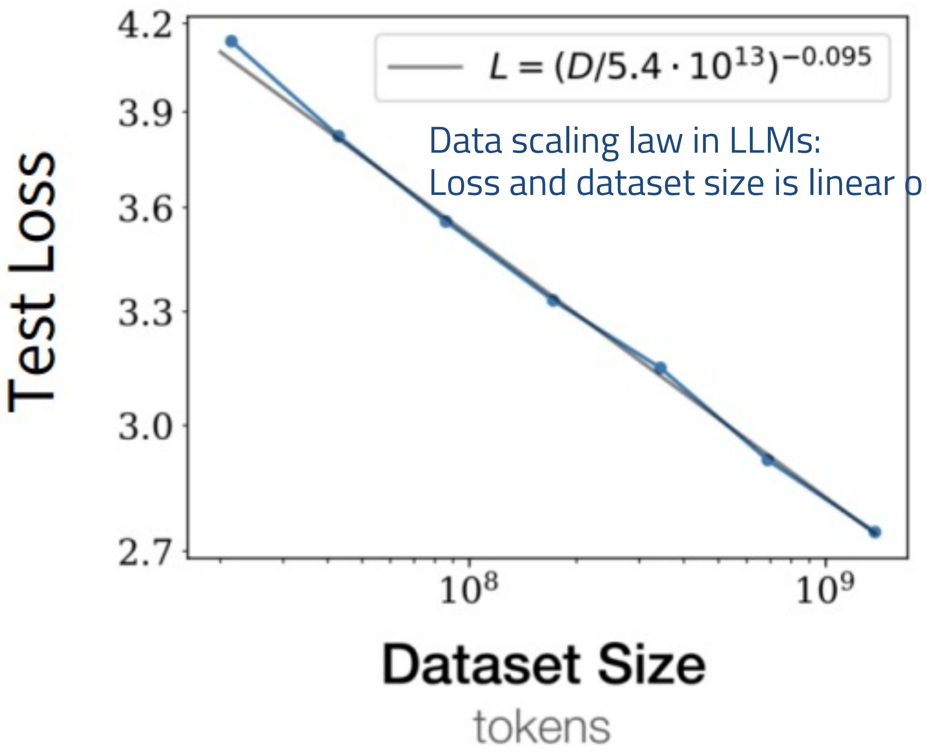


Data vs performance

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



(Hestness+ 2017)



Data scaling law in LLMs:
Loss and dataset size is linear on a log-log plot

(Kaplan+ 2020)

Kaplan et al. (2020)

Scaling Laws

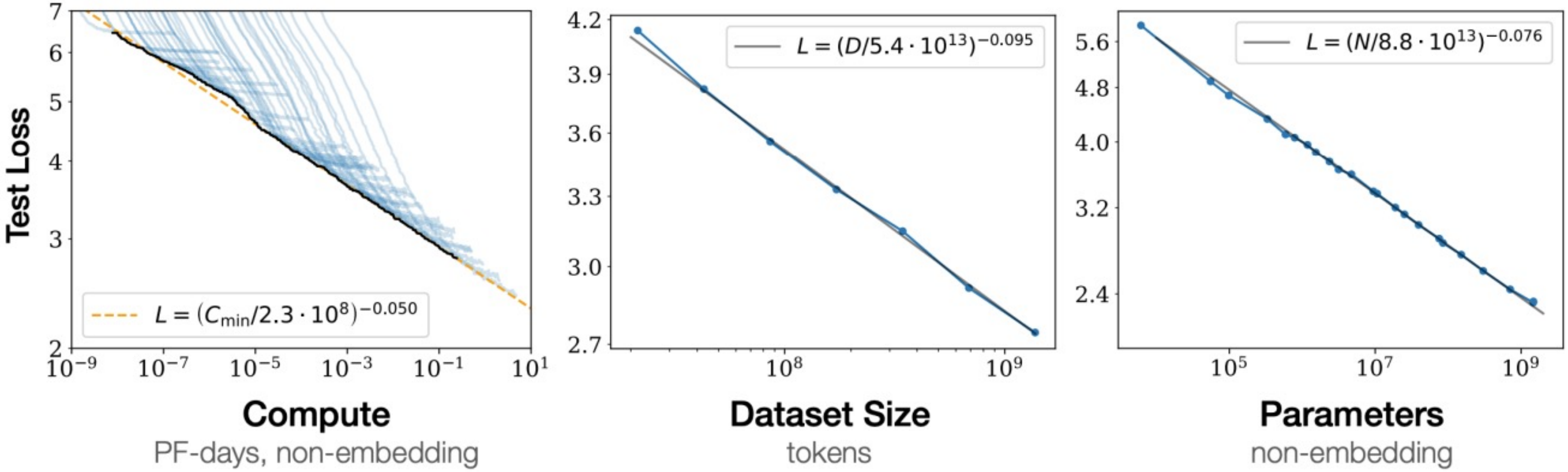
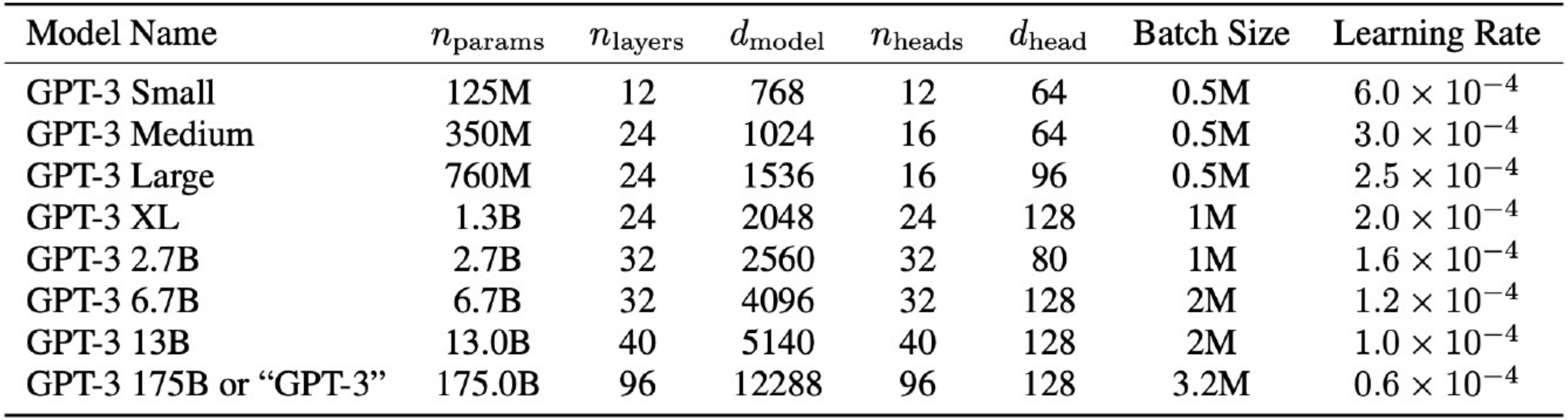


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Brown et al. (2020)

GPT3

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

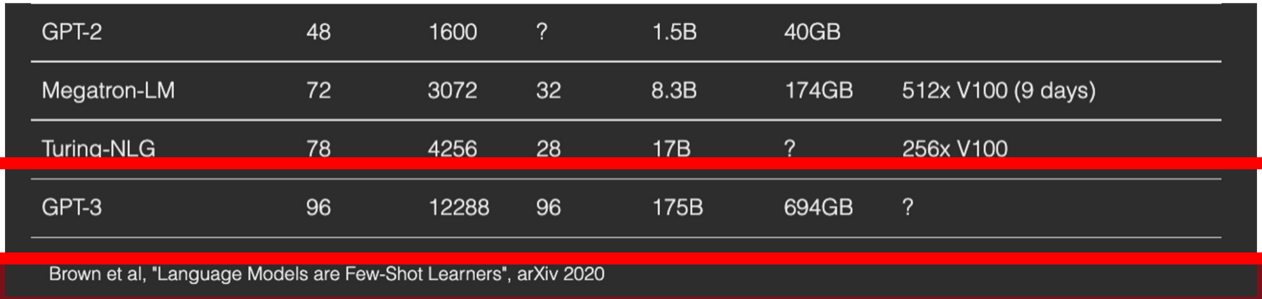


Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	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):

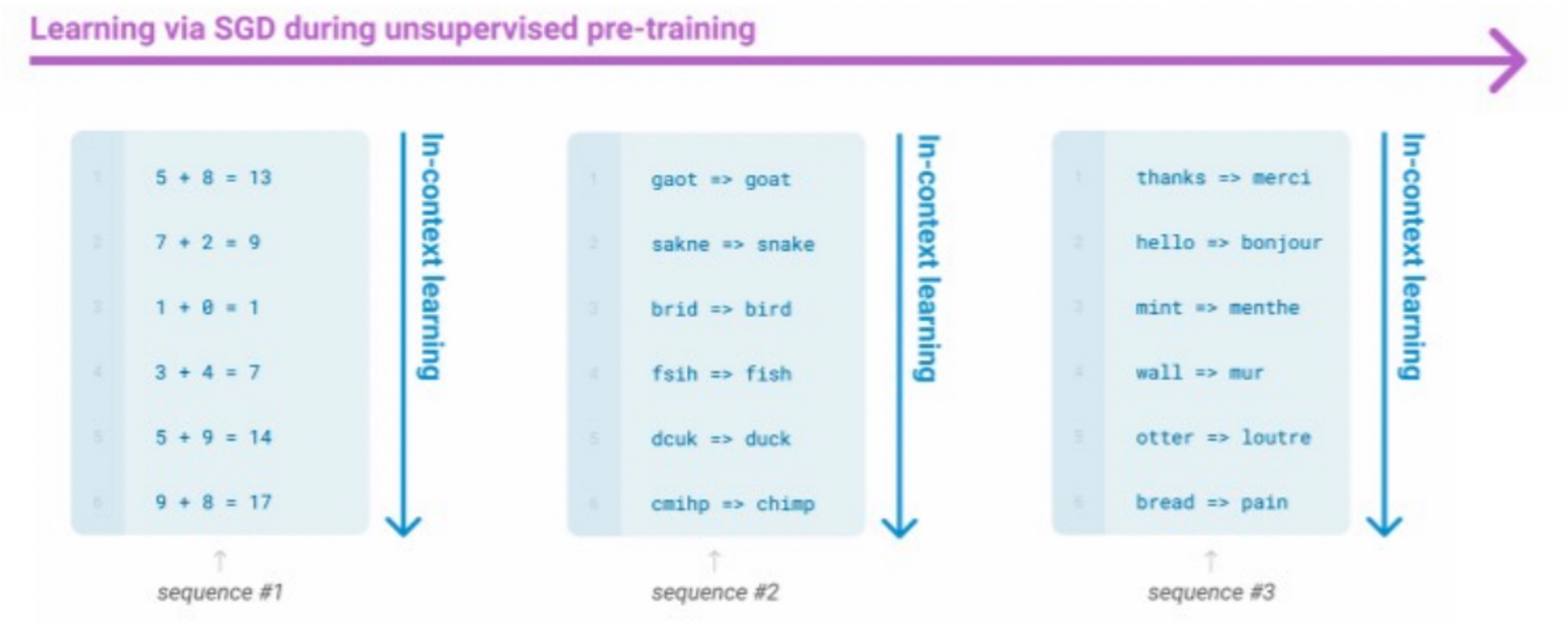
“
thanks -> merci
hello -> bonjour
mint -> menthe
otter ->
”

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.



Learning via SGD during unsupervised pre-training

5 + 8 = 13
7 + 2 = 9
1 + 0 = 1
3 + 4 = 7
5 + 9 = 14
9 + 8 = 17

In-context learning

sequence #1

gaot => goat
sakne => snake
brid => bird
fsih => fish
dcuk => duck
cmihp => chimp

In-context learning

sequence #2

thanks => merci
hello => bonjour
mint => menthe
wall => mur
otter => loutre
bread => pain

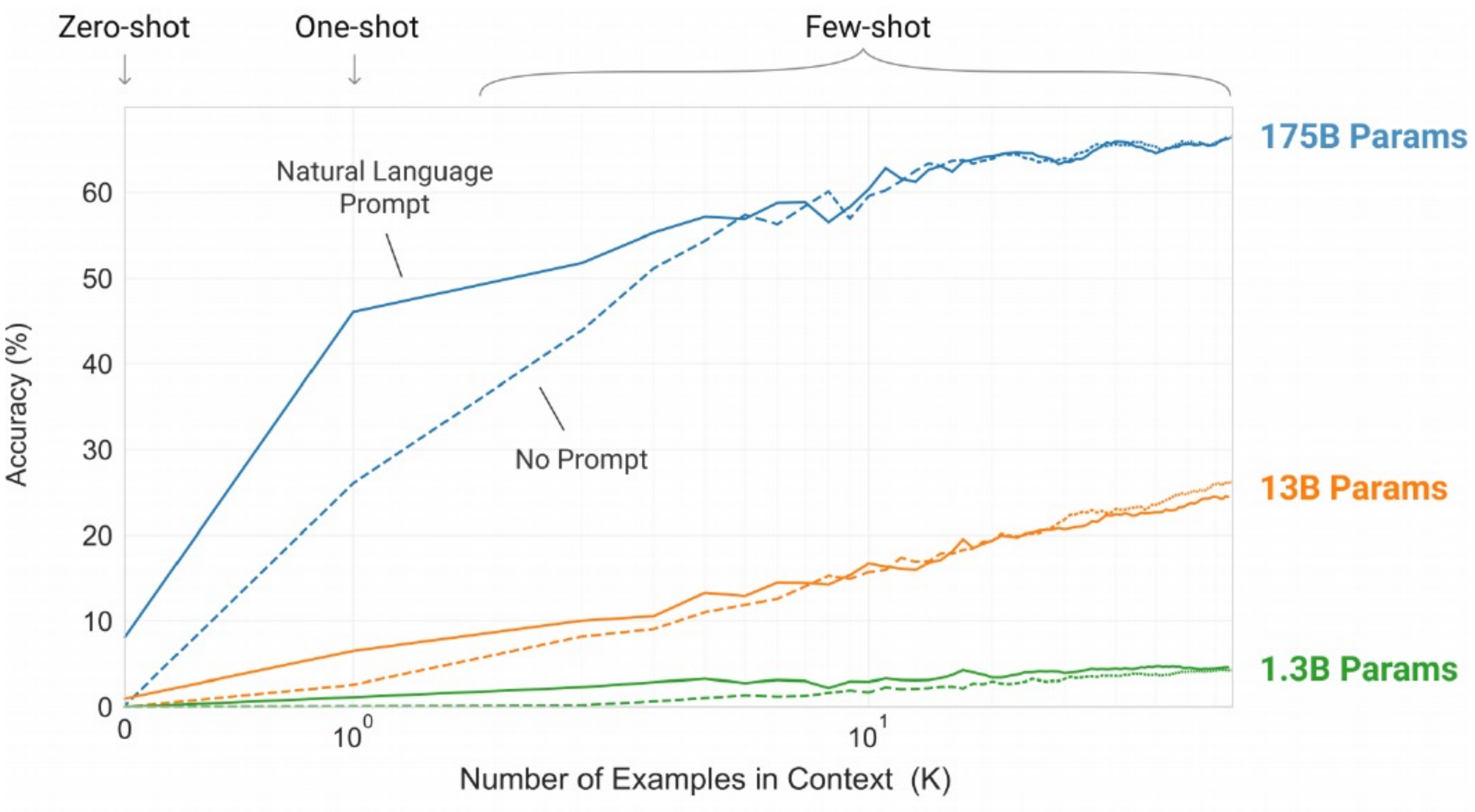
In-context learning

sequence #3

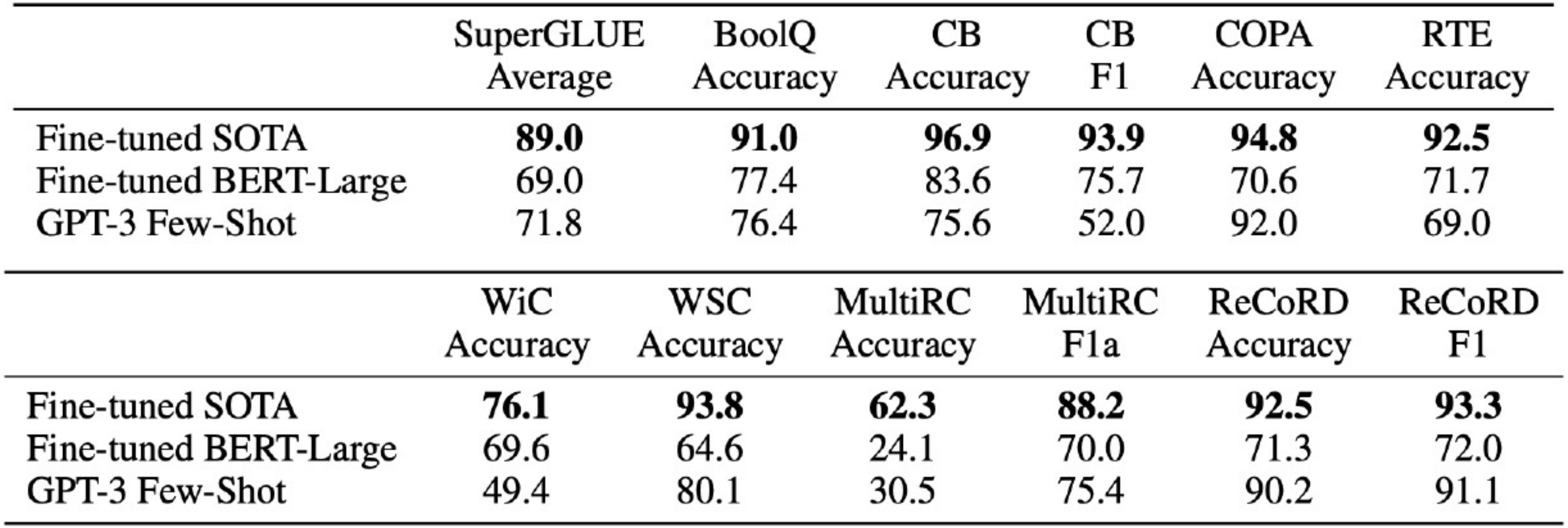
Brown et al. (2020)

GPT3

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



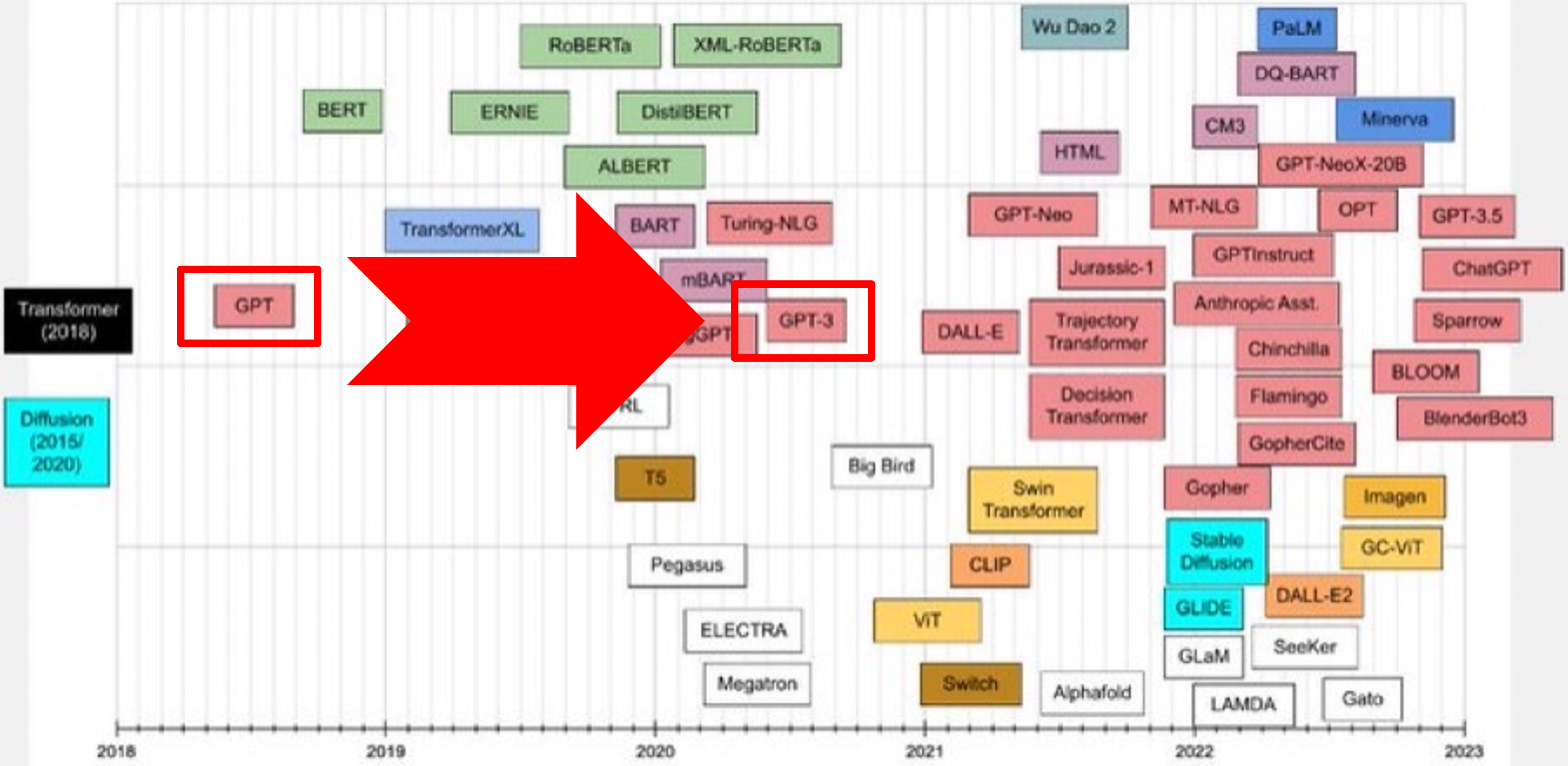
GPT3



	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!


Transformer (2018)
Diffusion (2015/ 2020)
GPT
BERT
RoBERTa
XML-RoBERTa
ERNIE
DistilBERT
ALBERT
TransformerXL
BART
Turing-NLG
mBART
GPT
GPT-3
RL
T5
Pegasus
ELECTRA
Megatron
Big Bird
Wu Dao 2
HTML
GPT-Neo
Jurassic-1
DALL-E
Trajectory Transformer
Decision Transformer
Swin Transformer
CLIP
ViT
Switch
Alphafold
PaLM
DQ-BART
CM3
Minerva
GPT-NeoX-20B
MT-NLG
OPT
GPT-3.5
GPTInstruct
ChatGPT
Anthropic Asst.
Sparrow
Chinchilla
BLOOM
Flamingo
BlenderBot3
GopherCite
Gopher
Imagen
Stable Diffusion
GC-ViT
GLIDE
DALL-E2
GLaM
SeeKer
LAMDA
Gato
2018
2019
2020
2021
2022
2023

Scaling law in language model

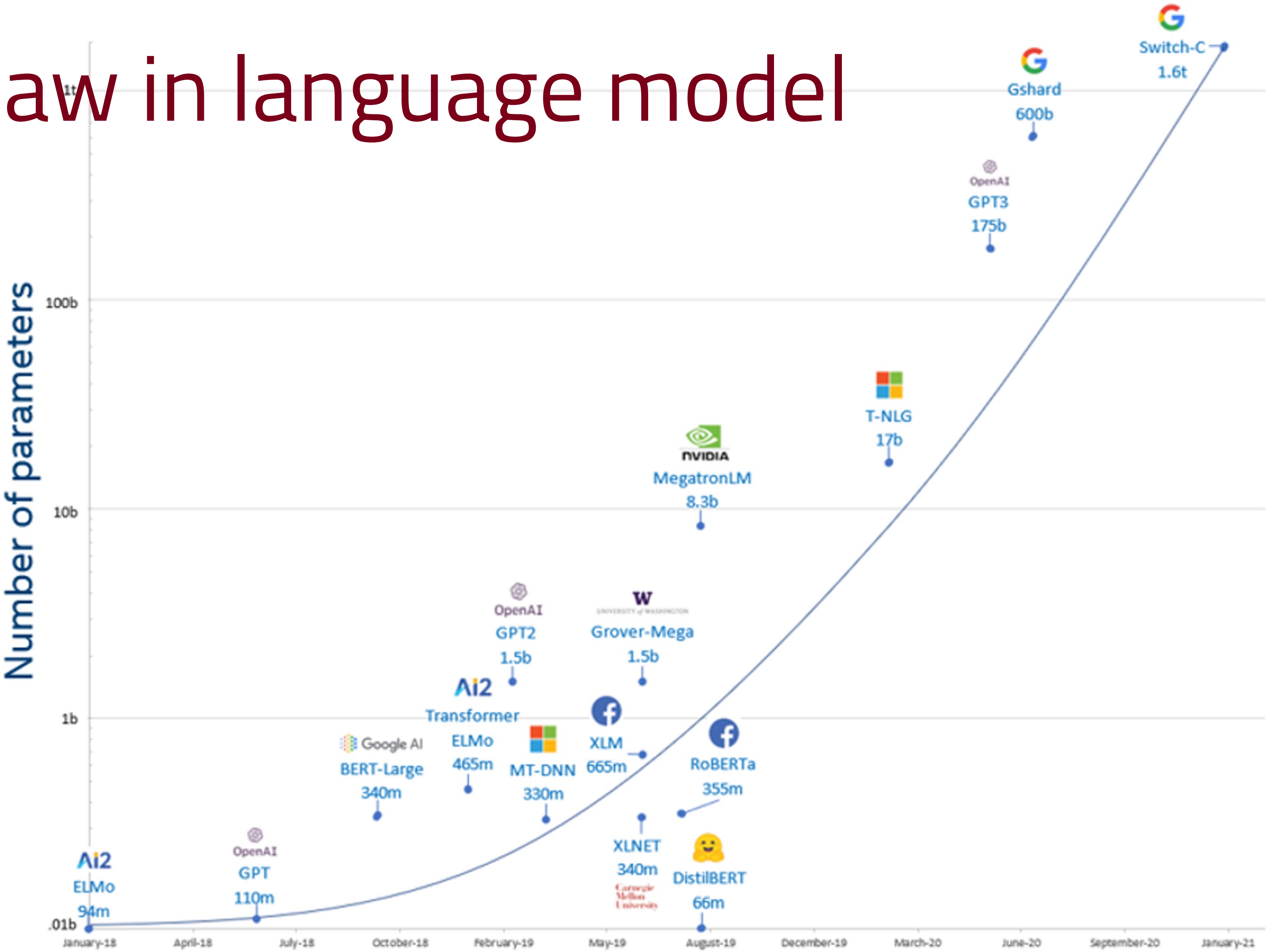
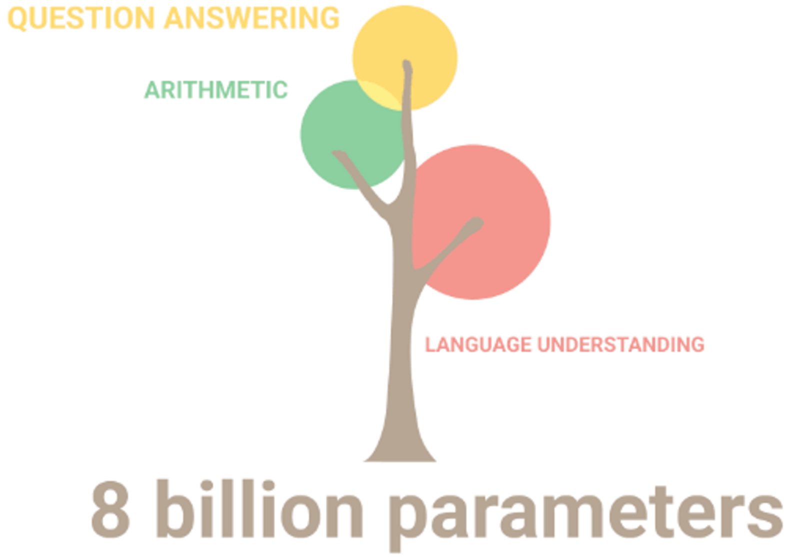
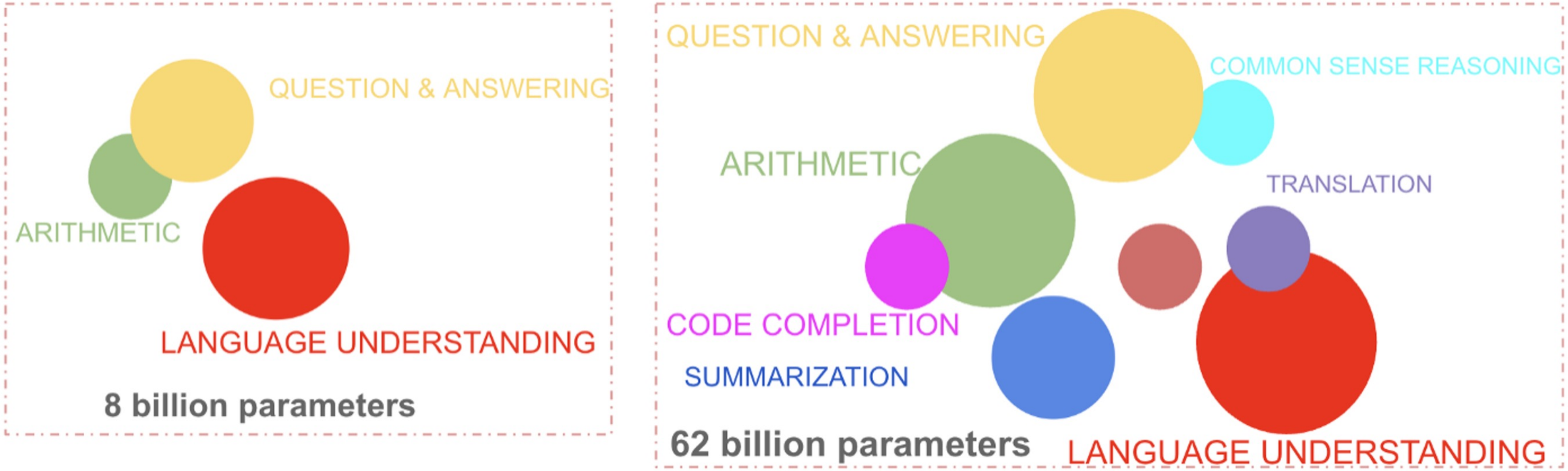


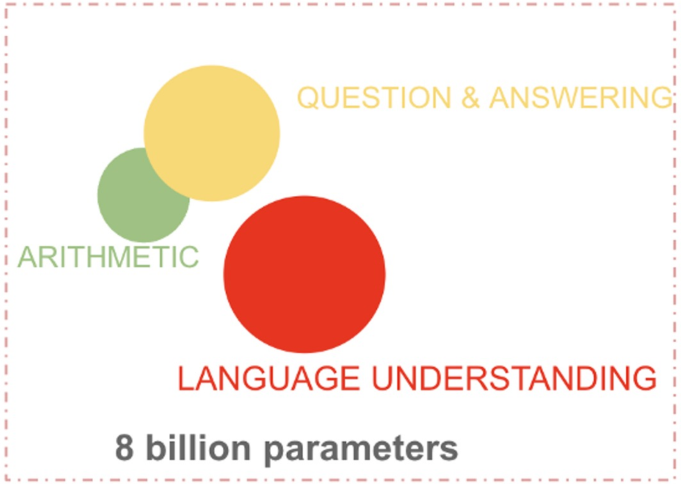
Figure 1: Exponential growth of number of parameters in DL models

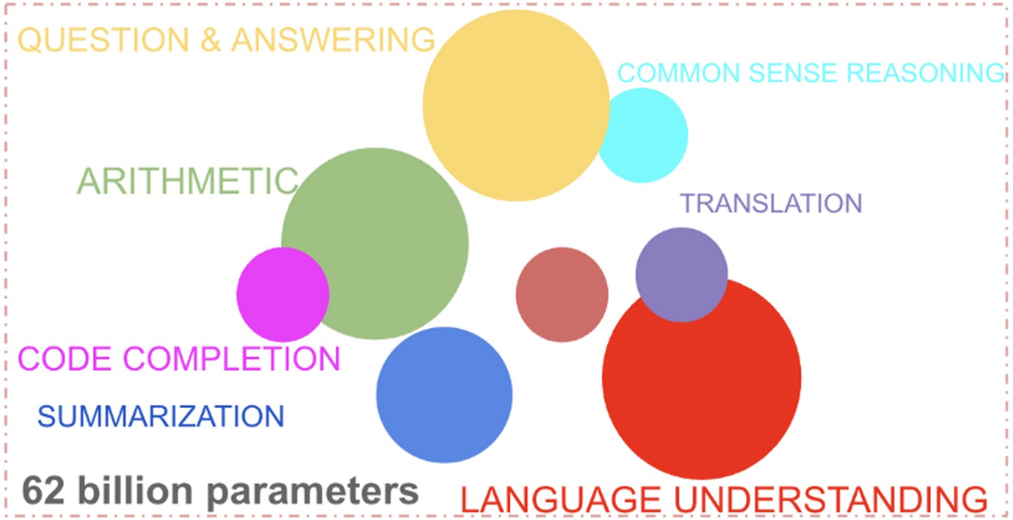


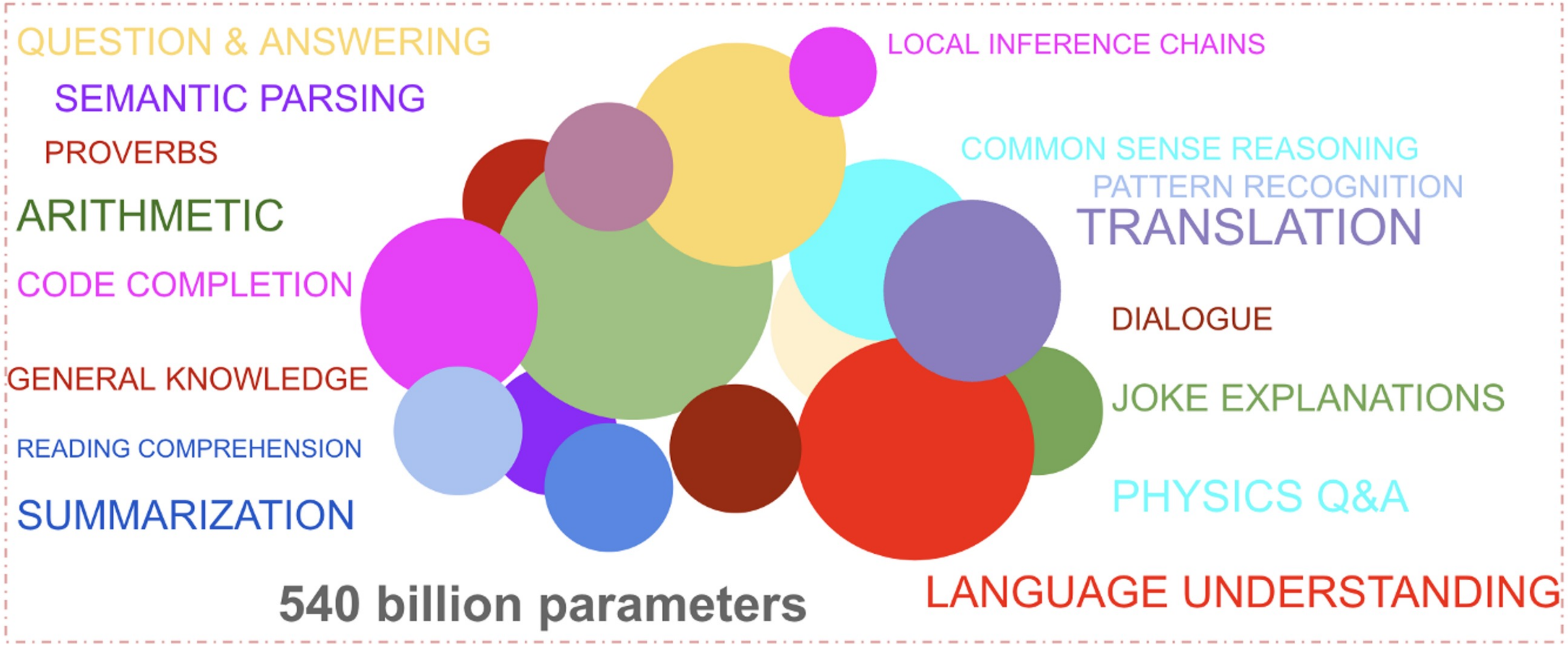
https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html



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



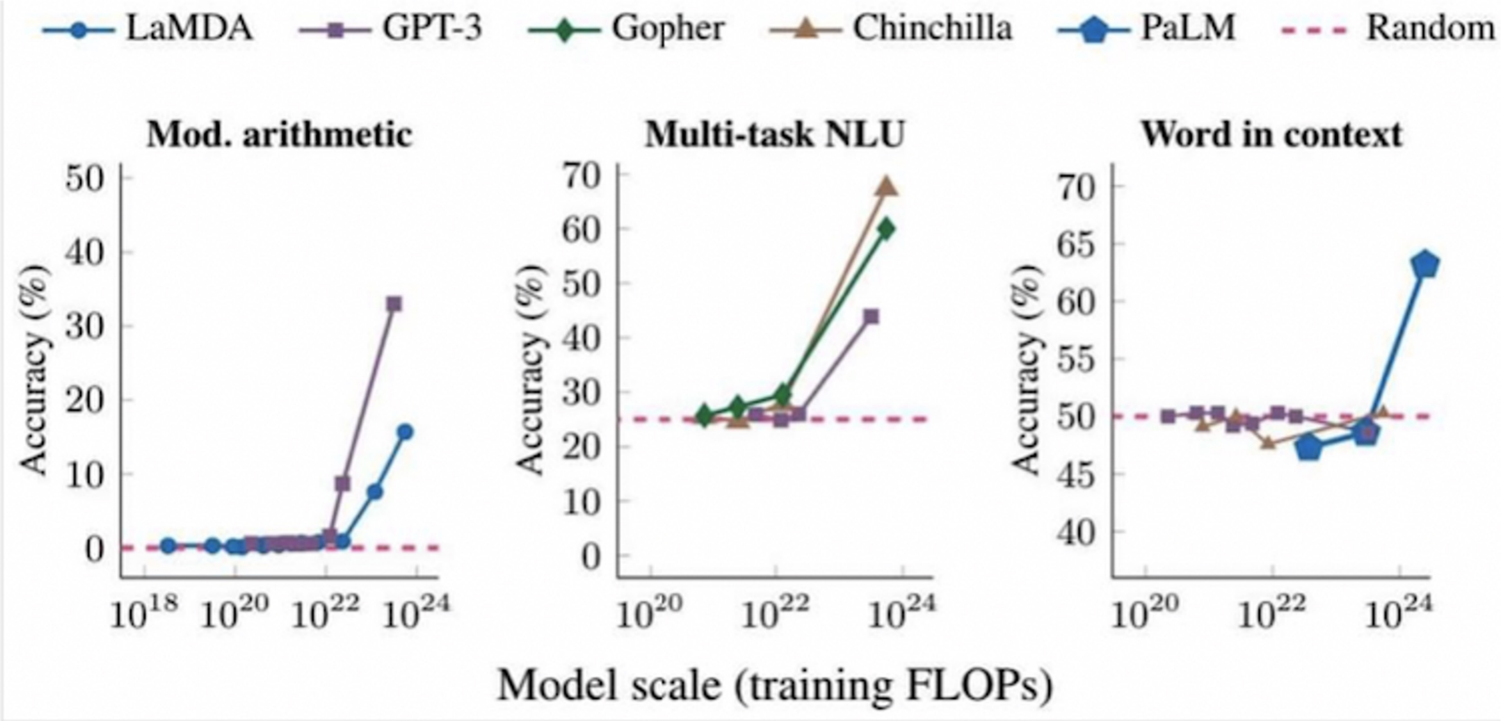




https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

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

Scaling Law in Vision-Language Model

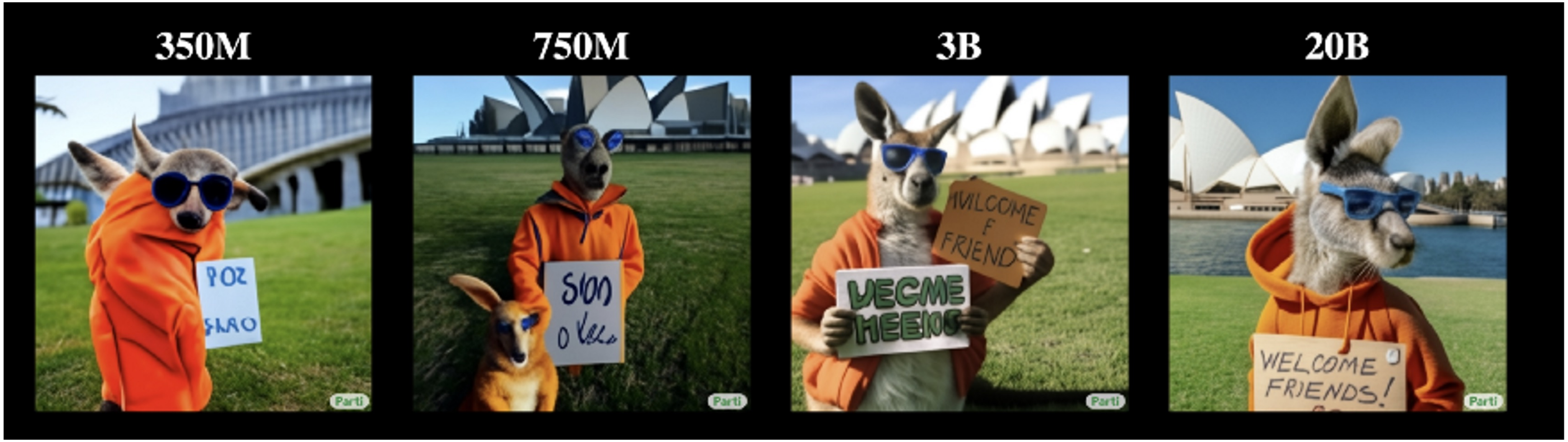


Figure 4. The generated image for the text "*A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!*". *Note the model gets the text in the image "welcome friends" correct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

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

Pre-Training Cost (with Google/AWS)

- ❑ BERT: Base $500, Large $7000
- ❑ Grover-MEGA: $25,000
- ❑ XLNet (BERT variant): $30,000 — $60,000 (unclear)

- ❑ This is for a single pre-training run…developing new pre-training techniques may require many runs
- ❑ Fine-tuning these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets)

hOps://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

Pre-Training Cost (with Google/AWS)

- GPT-3: estimated to be $4.6M.
  - One recent estimate pegged the cost of running GPT-3 on a single AWS web server to cost $87,000 a year at minimum
  - This cost has a large carbon footprint
    - Carbon footprint: equivalent to driving 700,000 km by car (source: Anthropocene magazine)
    - Counterpoints: GPT-3 isn't trained frequently, equivalent to 100 people traveling 7000 km for a conference, can use renewables

- BERT-Base pre-training: carbon emissions roughly on the same order as a single passenger on a flight from NY to San Francisco

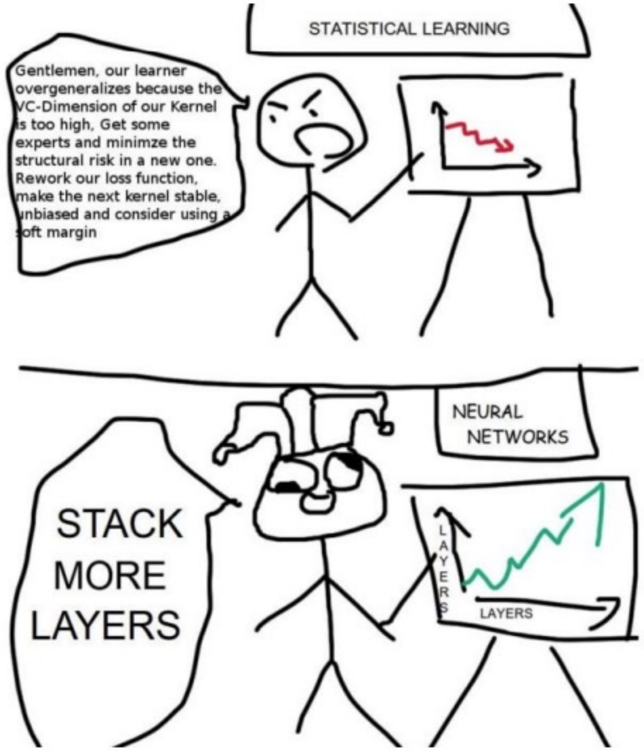
Strubell et al. (2019)

https://lambdalabs.com/blog/demysHfying-gpt-3/

https://www.technologyreview.com/2019/06/06/239031/training-a-singleai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifeHmes/

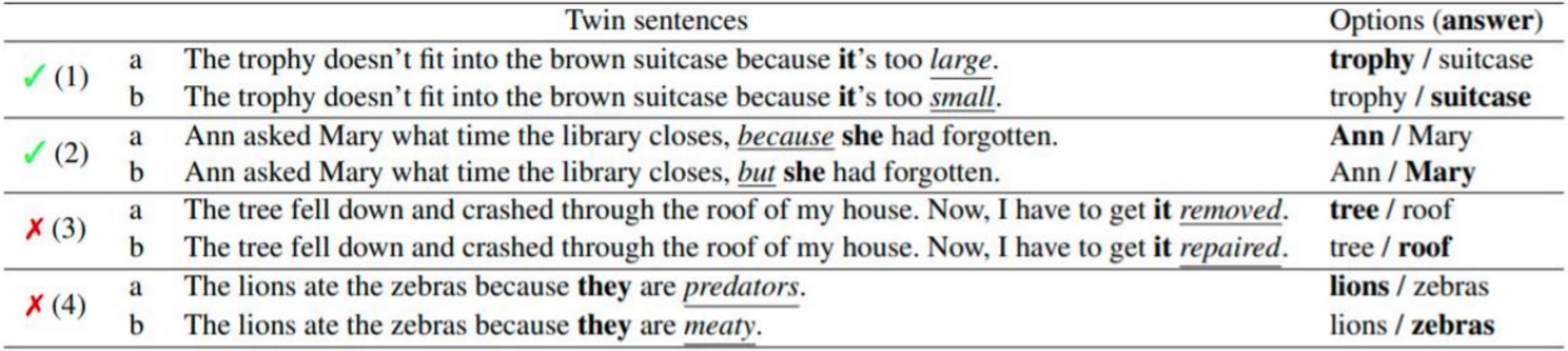
Q: Can big language models solve every problem?

- ❑ 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?


STATISTICAL LEARNING
Gentlemen, our learner overgeneralizes because the VC-Dimension of our Kernel is too high, Get some experts and minimze the structural risk in a new one. Rework our loss function, make the next kernel stable, unbiased and consider using a soft margin
NEURAL NETWORKS
STACK MORE LAYERS
LAYERS
LAYERS

Taken from r/programmerhumor

Will we solve the Winograd schema?

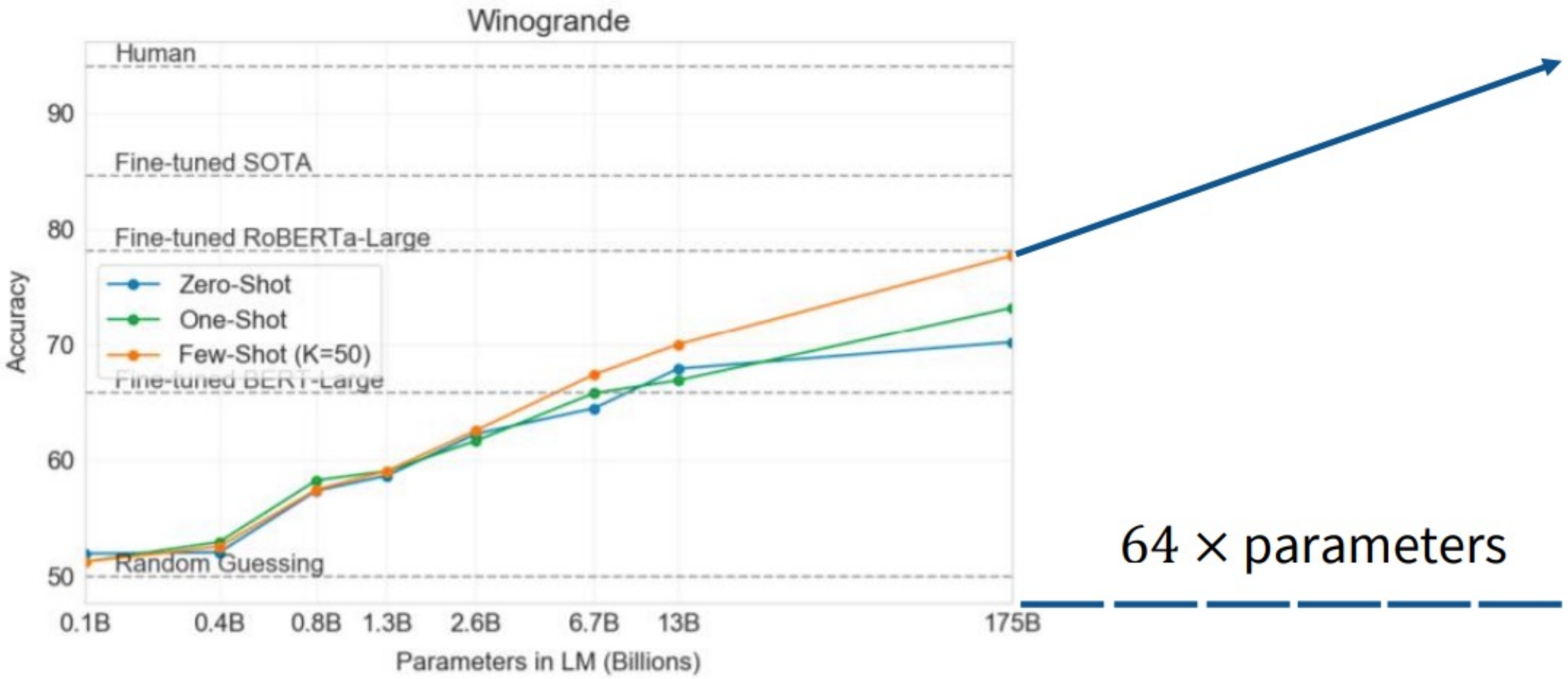


		Twin sentences	Options (**answer**)
✓ (1)	a	The trophy doesn't fit into the brown suitcase because **it**'s too *large*.	**trophy** / suitcase
	b	The trophy doesn't fit into the brown suitcase because **it**'s too *small*.	trophy / **suitcase**
✓ (2)	a	Ann asked Mary what time the library closes, *because* **she** had forgotten.	**Ann** / Mary
	b	Ann asked Mary what time the library closes, *but* **she** had forgotten.	Ann / **Mary**
✗ (3)	a	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *removed*.	**tree** / roof
	b	The tree fell down and crashed through the roof of my house. Now, I have to get **it** *repaired*.	tree / **roof**
✗ (4)	a	The lions ate the zebras because **they** are *predators*.	**lions** / zebras
	b	The lions ate the zebras because **they** are *meaty*.	lions / **zebras**

Current GPT-3 performance after seeing 50 examples: 77%. Can we push this further?

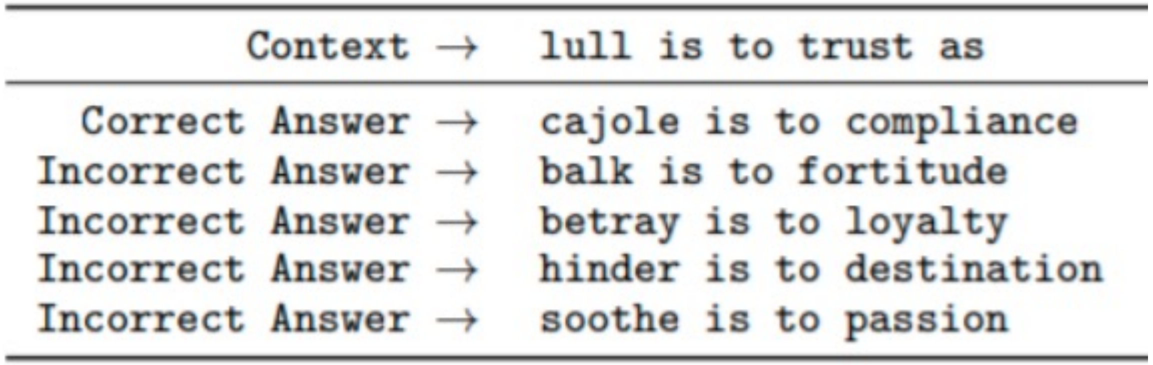
How much more compute for human-level reasoning?

Just extend the line for the scaling law..



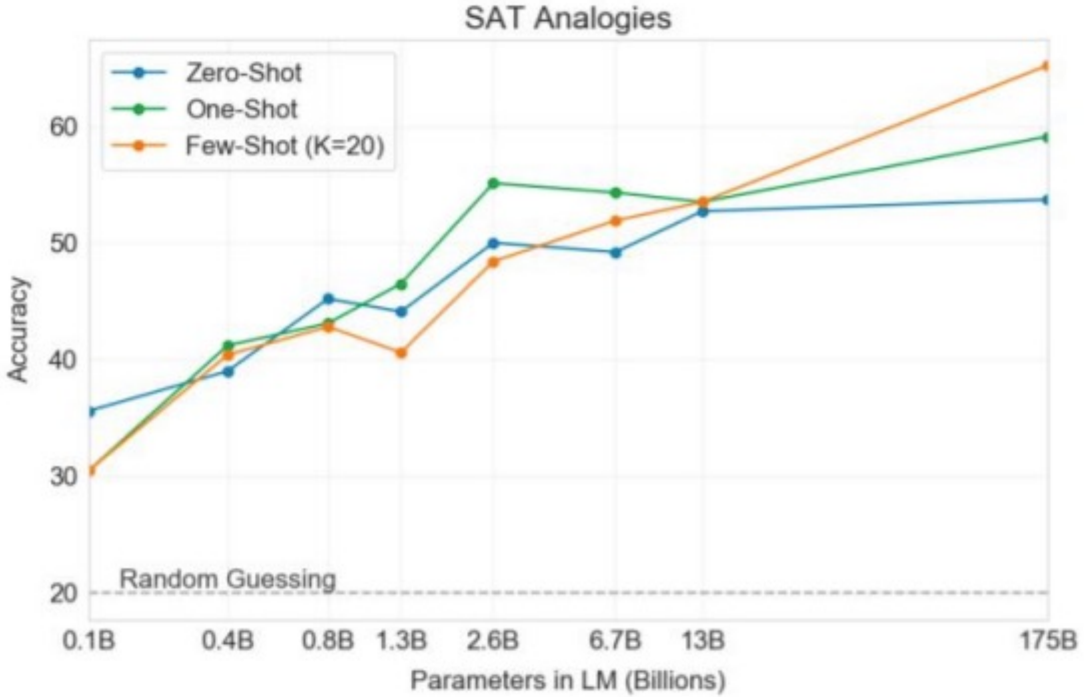
If the scaling law holds.. Roughly 64 times more parameters will get us to human-level

Another setting: SAT analogies

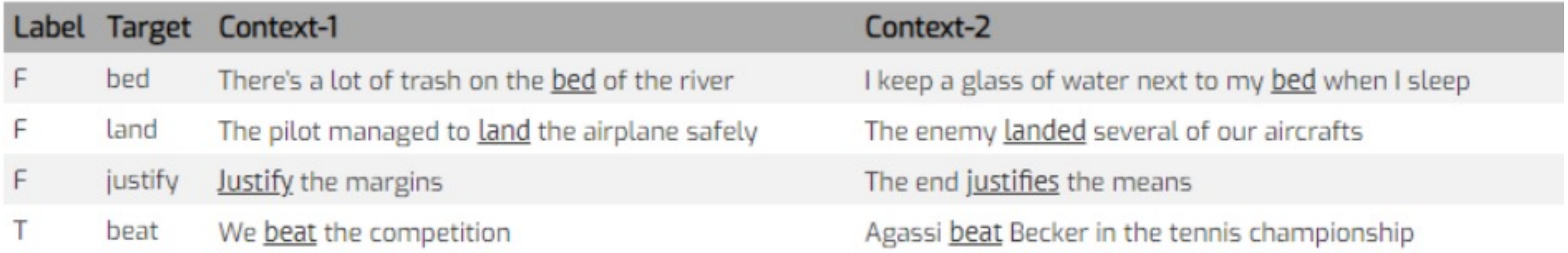


Context →	lull is to trust as
Correct Answer →	cajole is to compliance
Incorrect Answer →	balk is to fortitude
Incorrect Answer →	betray is to loyalty
Incorrect Answer →	hinder is to destination
Incorrect Answer →	soothe is to passion

- ❑ Scaling: clear linear scaling in log space.

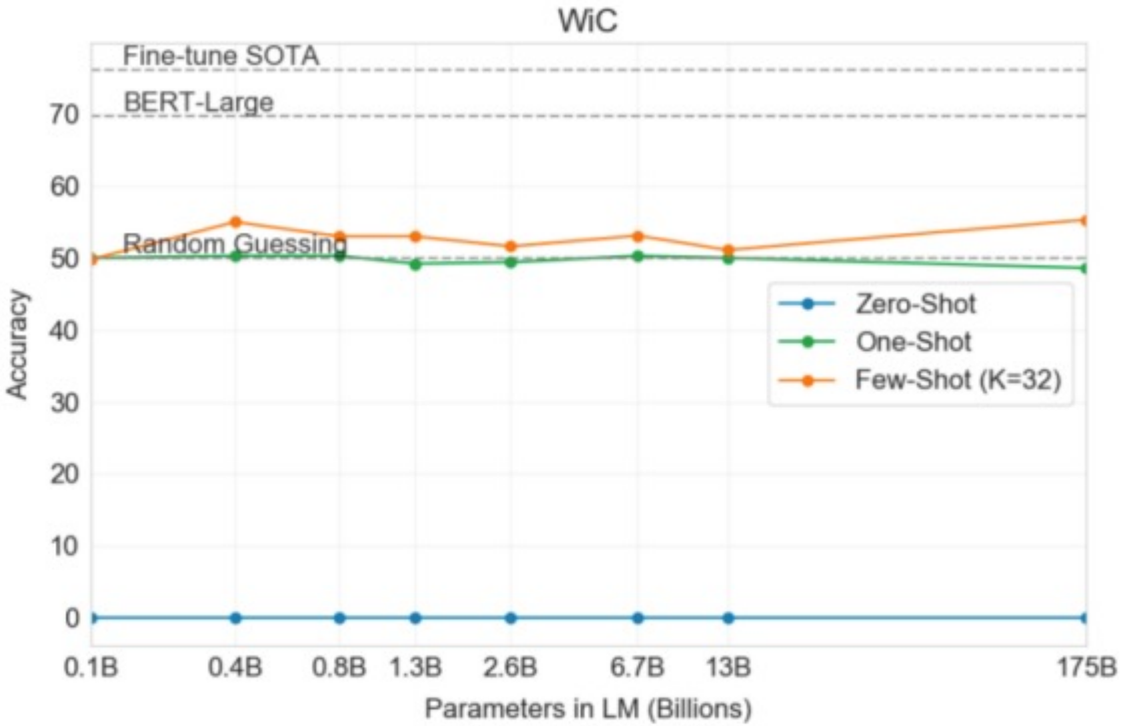


Less optimistic scaling curves



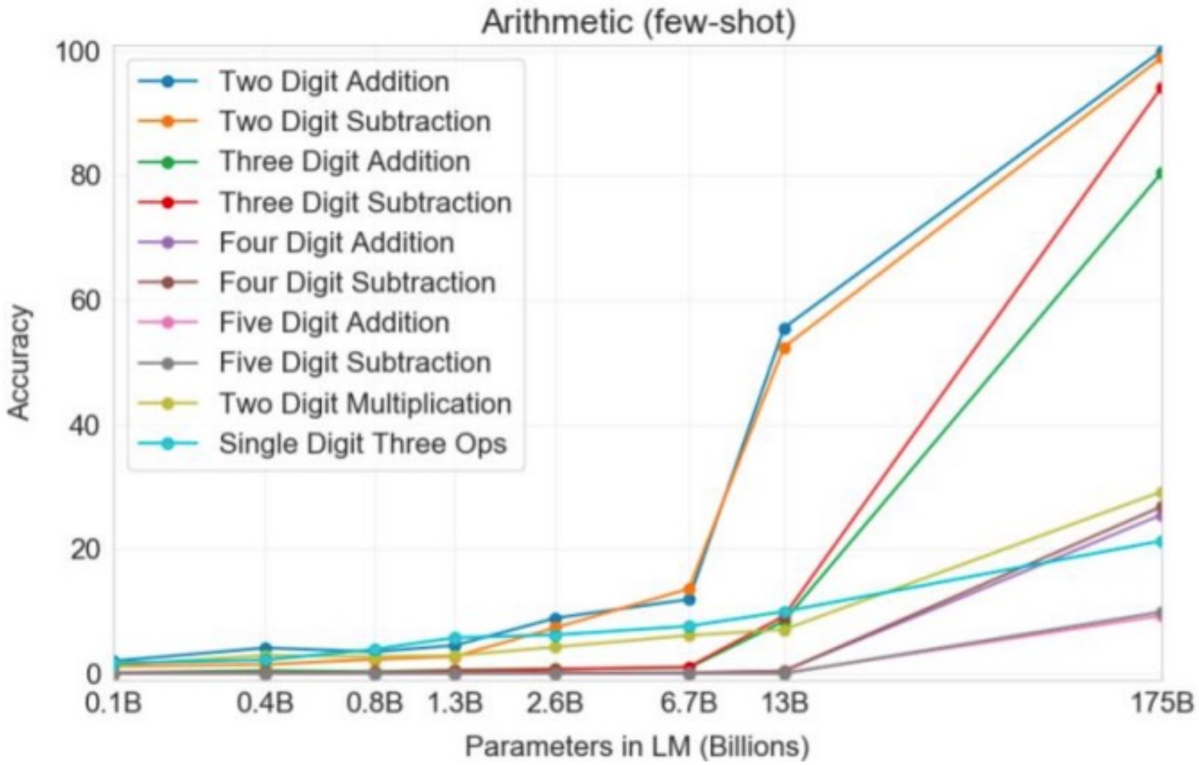
Label	Target	Context-1	Context-2
F	bed	There's a lot of trash on the bed of the river	I keep a glass of water next to my bed when I sleep
F	land	The pilot managed to land the airplane safely	The enemy landed several of our aircrafts
F	justify	Justify the margins	The end justifies the means
T	beat	We beat the competition	Agassi beat Becker in the tennis championship

- Scaling: near-zero. GPT-3 paper notes ‘pairwise comparison’ tasks are harder.



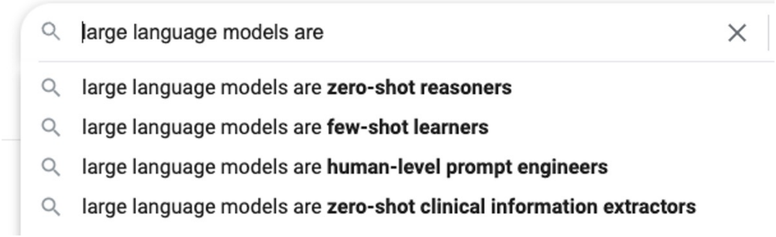
Phase transitions

- Thus far: everything has had linea 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!



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 quadratic jump (i.e., emergent behavior), but some fails.
- □ 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 issues left to explore!
  - ○ Bias, toxicity, and fairness
  - ○ Other capabilities such as reasoning, planning, knowledge base ..
  - ○ Grounding on robotics, vision, etc.



large language models are

large language models are **zero-shot reasoners**

large language models are **few-shot learners**

large language models are **human-level prompt engineers**

large language models are **zero-shot clinical information extractors**