北京航空航天大學
BEIHANG UNIVERSITY

Large Language Models

人工智能研究院

主讲教师 刘偲 沙磊 厍睿 郭晋阳

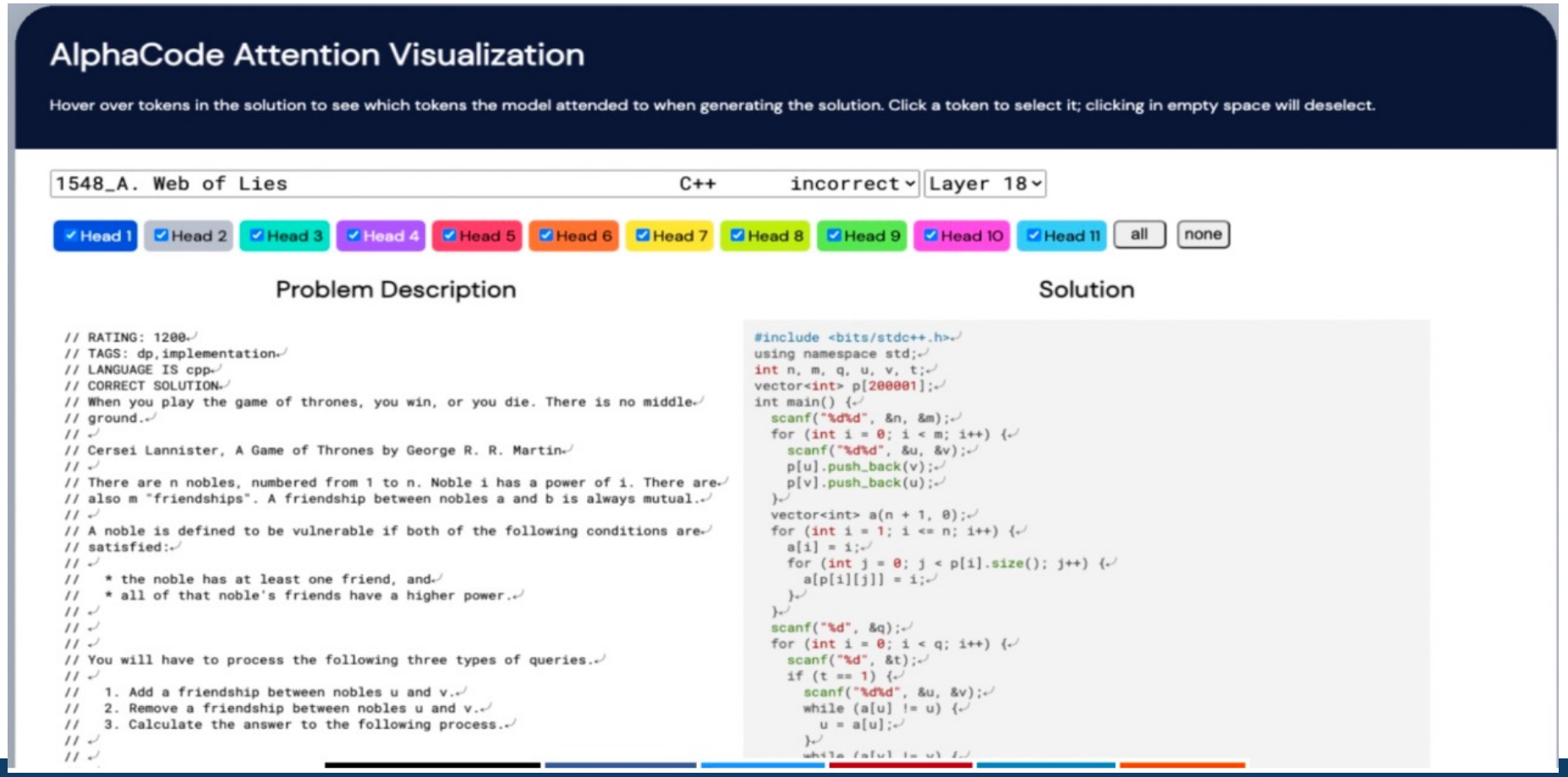
PLMs

Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - Very large models and in-context learning

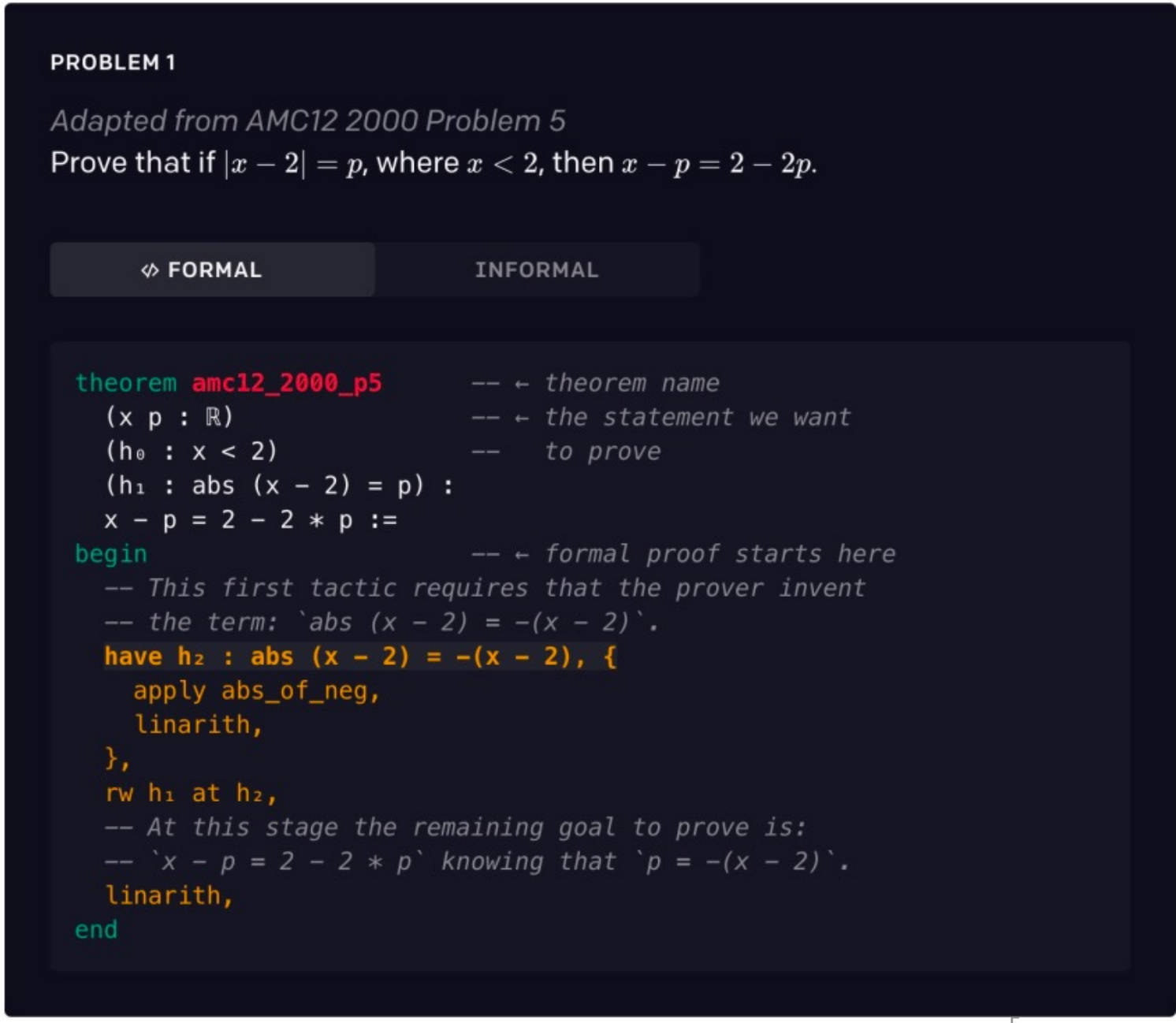
Breaking (Transformer) News!

- AlphaCode (a pre-trained Transformer-based code generation model) achieved a top 54.3% rating on Codeforces programming competitions!

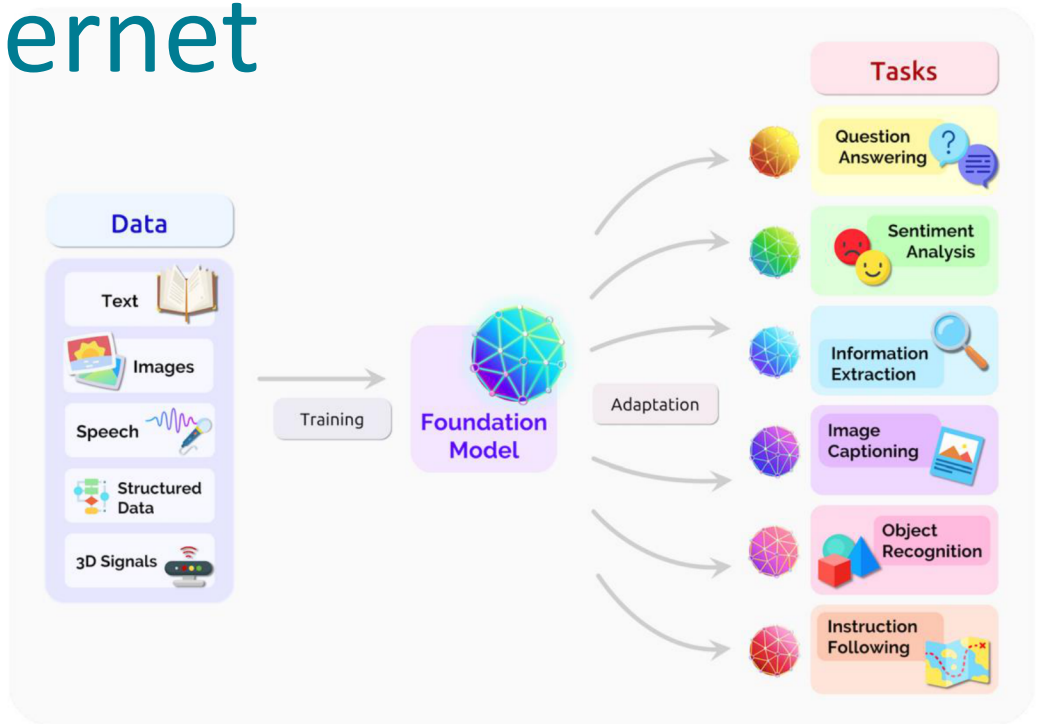


More Breaking (Transformer) News!

- Pre-Trained Transformer-Based theorem prover sets new state-of-the-art (41.2% vs. 29.3%) on a collection of challenging math Olympiad questions (miniF2F) [Polu et al., 2022]



Pretraining – scaling unsupervised learning on the internet



Key ideas in pretraining

- Make sure your model can process large-scale, diverse datasets
- Don't use labeled data (otherwise you can't scale!)
- Compute-aware scaling

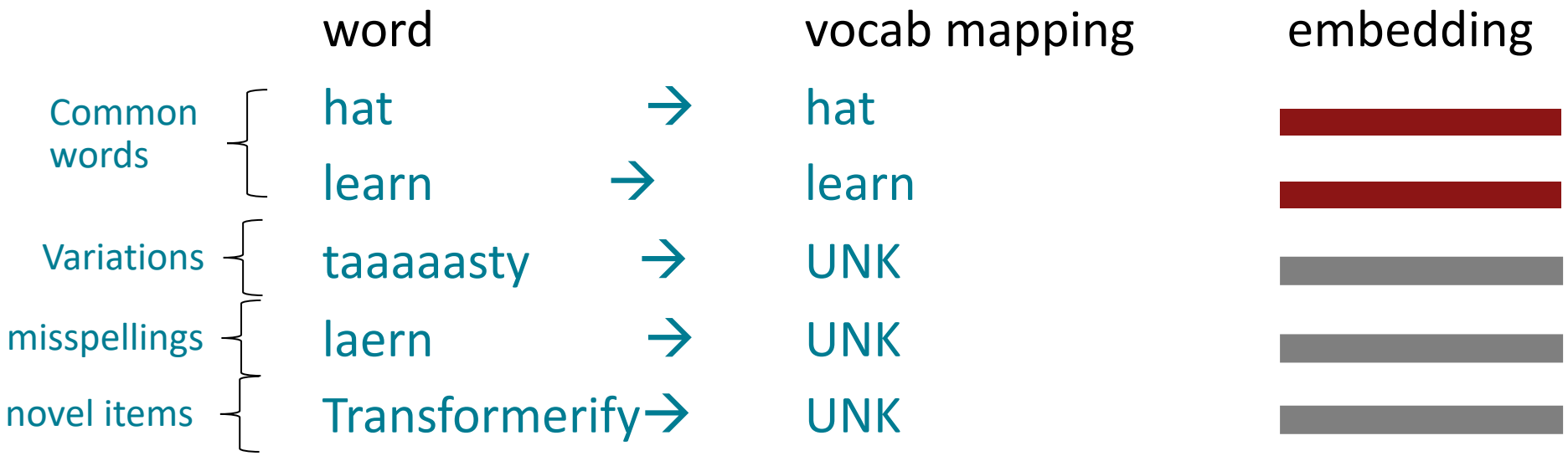
Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - Very large models and in-context learning

Word structure and subword models

Let's take a look at the assumptions we've made about a language's vocabulary.

We assume a fixed vocab of tens of thousands of words, built from the training set.
All *novel* words seen at test time are mapped to a single UNK.



Word structure and subword models

Finite vocabulary assumptions make even *less* sense in many languages.

- Many languages exhibit complex **morphology**, or word structure.
  - The effect is more word types, each occurring fewer times.

Example: Swahili verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)

Here's a small fraction of the conjugations for *ambia* – to tell.

Conjugation of *-ambia* [less ▲]

Form	Positive	Negative
Infinitive	kuambia	kutoambia
Positive form	Singular	Plural
Imperative	ambia	ambieni
Habitual	huambia	

Polarity	1st Sg.	1st Pl.	2nd Sg.	2nd Pl.	3rd / M-wa Sg. / 1	3rd / M-wa Pl. / 2	M-mi 3	M-mi 4	Ma 5	Ma 6	Ki-vi 7	Ki-vi 8	N 9	N 10	U 11 / 14	Ku 15 / 17	Pa 16	Mu 18
Past [less ▲]																		
Positive	niliambia naliambia	tuliambia twaliambia	uliambia waliambia	mliambia mwaliambia	aliambia	waliambia	uliambia	iliambia	liliambia	yaliambia	kiliambia	viliambia	iliambia	ziliambia	uliambia	kuliambia	paliambia	muliambia
Negative	sikuambia	hatukuambia	hukuambia	hamkuambia	hakuambia	hawakuambia	haukuambia	haikuambia	halikuambia	hayakuambia	hakikuambia	havikuambia	haikuambia	hazikuambia	haukuambia	hakukuambia	hapakuambia	hamukuambia
Present [less ▲]																		
Positive	ninaambia naambia	tunaambia	unaambia	mnaambia	anaambia	wanaambia	unaambia	inaambia	linaambia	yanaambia	kinaambia	vinaambia	inaambia	zinaambia	unaambia	kunaambia	panaambia	munaambia
Negative	siambii	hatuambii	huambii	hamambii	haambii	hawaambii	hauambii	haiambii	haliambii	hayaambii	hakiambii	haviambii	haiambii	haziambii	hauambii	hakuambii	hapaambii	hamuambii
Future [less ▲]																		
Positive	nitaambia	tutaambia	utaambia	mtaambia	ataambia	wataambia	utaambia	itaambia	litaambia	yataambia	kitaambia	vitaambia	itaambia	zitaambia	utaambia	kutaambia	pataambia	mutaambia
Negative	sitaambia	hatutaambia	hutaambia	hamtaambia	hataambia	hawataambia	hautaambia	haitaambia	halitaambia	hayataambia	hakitaambia	havitaambia	haitaambia	hazitaambia	hautaambia	hakutaambia	hapataambia	hamutaambia
Subjunctive [less ▲]																		
Positive	niambie	tuambie	uambie	mambie	aambie	waambie	uambie	iambie	liambie	yaambie	kiambie	viambie	iambie	ziambie	uambie	kuambie	paambie	muambie
Negative	nisiambie	tusiambie	usiambie	msiambie	asiambie	wasiambie	usiambie	isiambie	lisiambie	yasiambie	kisiambie	visiambie	isiambie	zisiambie	usiambie	kusiambie	pasiambie	musiambie
Present Conditional [less ▲]																		
Positive	ningeambia	tungeambia	ungeambia	mngeambia	angeambia	wangeambia	ungeambia	ingeambia	lingeambia	yangeambia	kingeambia	vingeambia	ingeambia	zingeambia	ungeambia	kungeambia	pangeambia	mungeambia
Negative	nisingeambia singeambia	tusingeambia hatungeambia	usingeambia hungeambia	msingeambia hamngeambia	asingeambia hangeambia	wasingeambia hawangeambia	usingeambia haungeambia	isingeambia haingeambia	lisingeambia halingeambia	yasingeambia hayangeambia	kisingeambia hakingeambia	visingeambia havingeambia	isingeambia haingeambia	zisingeambia hazingeambia	usingeambia haungeambia	kusingeambia hakungeambia	pasingeambia hapangeambia	musingeambia hamungeambia
Past Conditional [less ▲]																		
Positive	ningaliambia	tungaliambia	ungaliambia	mngaliambia	angaliambia	wangaliambia	ungaliambia	ingaliambia	lingaliambia	yangaliambia	kingaliambia	vingaliambia	ingaliambia	zingaliambia	ungaliambia	kungaliambia	pangaliambia	mungaliambia
Negative	nisingaliambia singaliambia	tusingaliambia hatungaliambia	usingaliambia hungaliambia	msingaliambia hamngaliambia	asingaliambia hangaliambia	wasingaliambia hawangaliambia	usingaliambia haungaliambia	isingaliambia haingaliambia	lisingaliambia halingaliambia	yasingaliambia hayangaliambia	kisingaliambia hakingaliambia	visingaliambia havingaliambia	isingaliambia haingaliambia	zisingaliambia hazingaliambia	usingaliambia haungaliambia	kusingaliambia hakungaliambia	pasingaliambia hapangaliambia	musingaliambia hamungaliambia
Conditional Contrary to Fact [less ▲]																		
Positive	ningeliambia	tungeliambia	ungeliambia	mngeliambia	angeliambia	wangeliambia	ungeliambia	ingeliambia	lingeliambia	yangeliambia	kingeliambia	vingeliambia	ingeliambia	zingeliambia	ungeliambia	kungeliambia	pangeliambia	mungeliambia
Gnomic [less ▲]																		
Positive	naambia	twaambia	waambia	mwaambia	aambia	waambia	waambia	yaambia	laambia	yaambia	chaambia	vyaambia	yaambia	zaambia	waambia	kwaambia	paambia	mwaambia
Perfect [less ▲]																		

[Wiktionary]

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 (subwordtokens).**
- At training and testing time, each word is split into a sequence of known subwords.

Byte-pair encoding is a simple, effective strategy for defining a subwordvocabulary.

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.

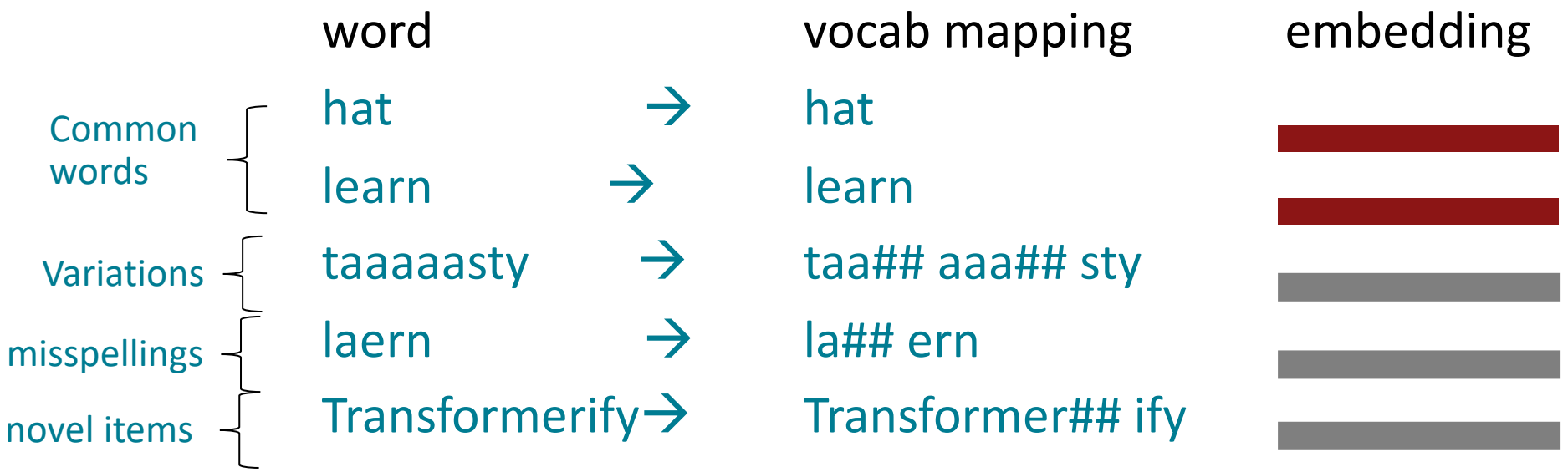
Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

[Sennrich et al., 2016, Wu et al., 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.



Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - Very large models and in-context learning

Motivating word meaning and context

Recall the adage we mentioned at the beginning of the course:

> *"You shall know a word by the company it keeps"*(J. R. Firth 1957: 11)

This quote is a summary of **distributional semantics**, and motivated **word2vec**. But:

> "… the complete meaning of a word is always contextual,
> and no study of meaning apart from a complete context
> can be taken seriously."(J. R. Firth 1935)

Consider *I* ***record****the* ***record***: the two instances of ***record***mean different things.

[Thanks to Yoav Goldberg on Twitter for pointing out the 1935 Firth quote.]

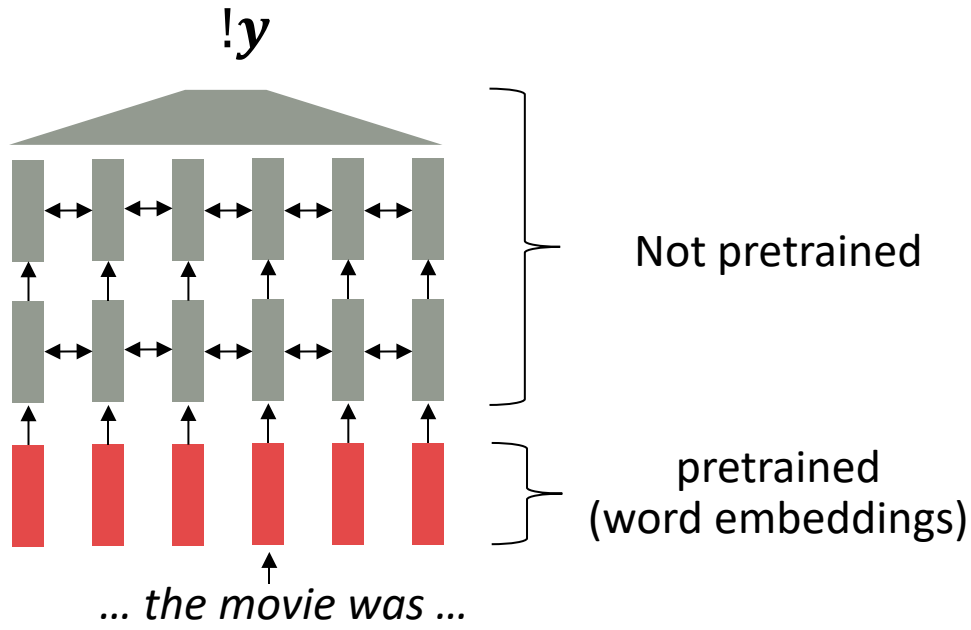
Where we were: **pretrained word embeddings**

Circa 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!



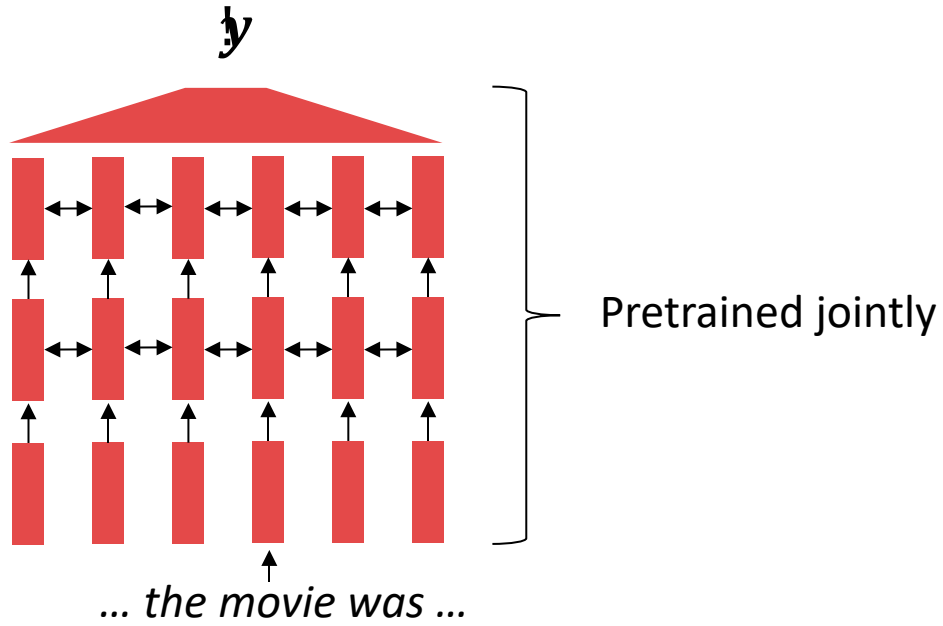
[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

Where we're going: **pretraining whole models**

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, and 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]

What can we learn from reconstructing the input?

- The woman walked across the street, checking for traffic over ___ shoulder.
- I went to the ocean to see the fish, turtles, seals, and _____.
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___.
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____

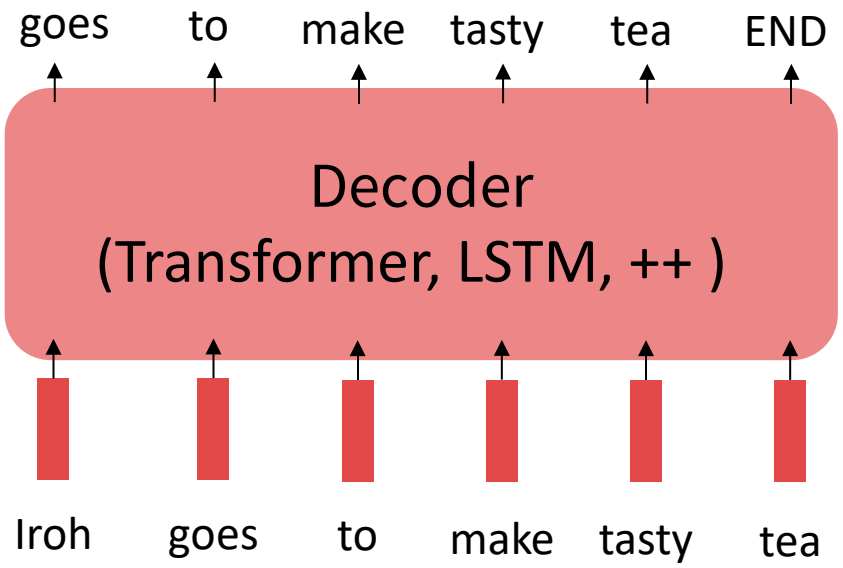
Pretraining through language modeling [Dai and Le, 2015]

Recall the **language modeling** task:

- Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.

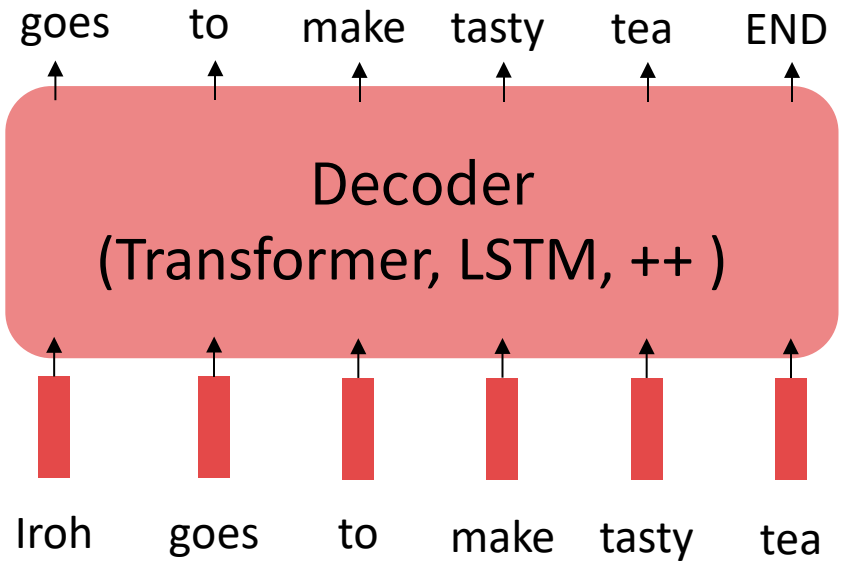


The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

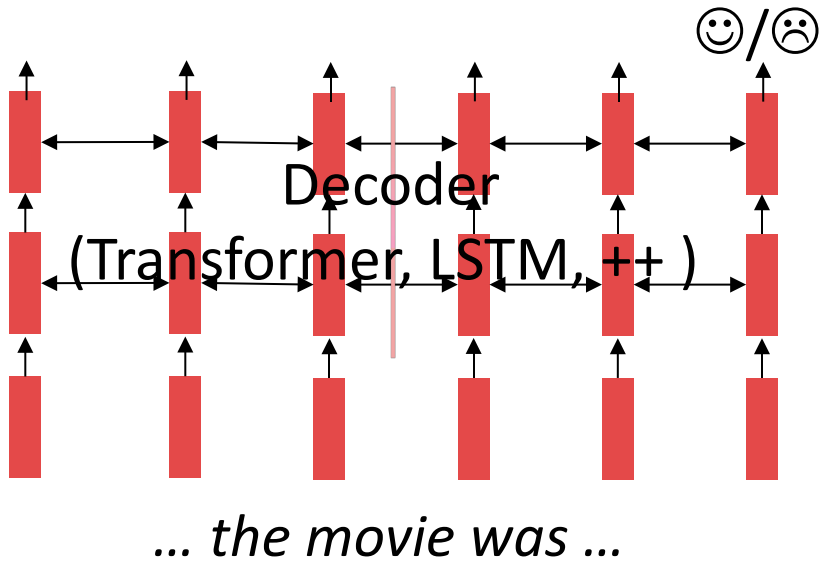
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



Stochastic gradient descent and pretrain/finetune

- Why should pretraining and finetuning help, from a "training neural nets" perspective?

- Consider, provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathcal{L}_{pretrain}(\theta)$
  - (The pretraining loss.)
- Then, finetuning approximates $\min_{\theta} \mathcal{L}_{finetune}(\theta)$, starting at $\hat{\theta}$
  - (The finetuning loss)
- The pretraining may matter because stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during finetuning.
  - So, maybe the finetuning local minima near $\hat{\theta}$ tend to generalize well!
  - And/or, maybe the gradients of finetuning loss near $\hat{\theta}$ propagate nicely!

Why unsupervised learning? Why not QA?

- Orders of magnitude difference in data size – there is a lot of high-quality text

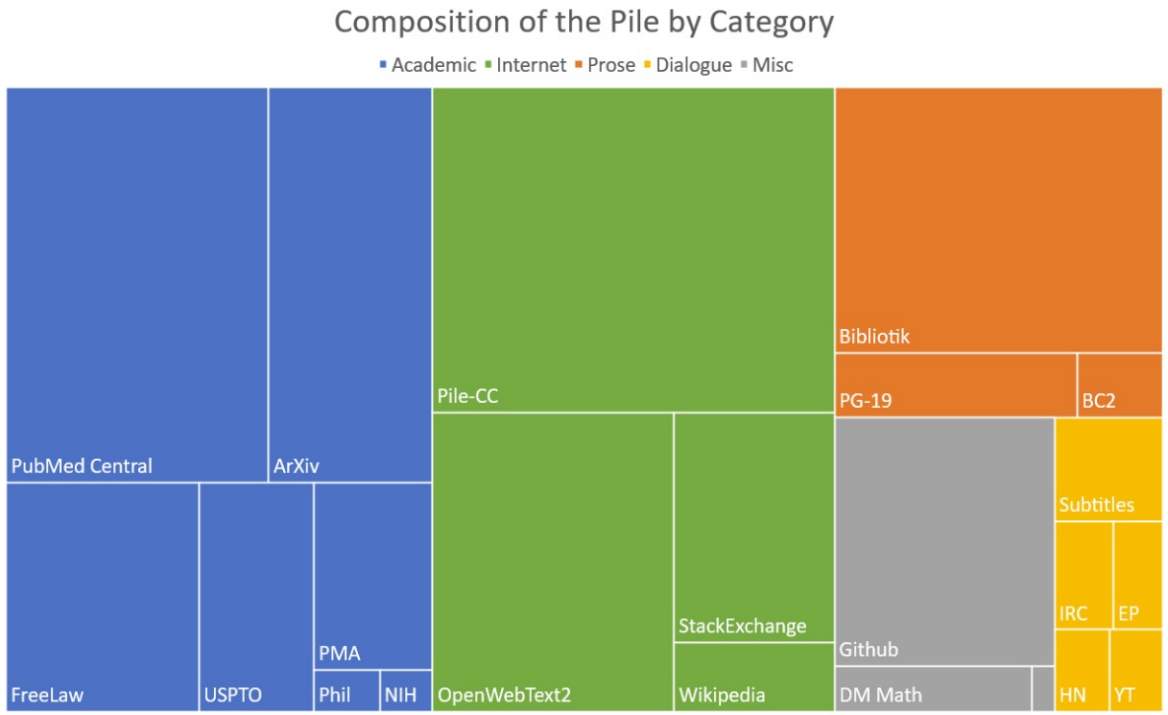
Dataset	Tokens(~0.75 words)
SQuAD 2.0 [Rajpukar+ 2018]	< 50 Million
DCLM-pool [Li+ 2024]	240 Trillion
Estimated ‘internet text’ [Villalobos 2024]	510T (indexed), 3100T (total)

A *10 million* times gap in QA to indexed internet

With this much data, we might make progress on even the hardest fill-in-the-blank tasks

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]

This gives us some weak coverage over an enormous range of downstream tasks

Pretraining data samples 1 [DCLM]

- Bizarro Wonder Woman is a bizarro version of Wonder Woman.\n\nWhen Bizarro III found himself infused with radiation from a blue sun, he developed the ability to replicate himself as well as create other \"Bizarro\" lifeforms based upon likenesses of people from Earth. He used this power to populate a cube-shaped planetoid dubbed Bizarro World within the blue sun star system. One of the many duplicates that he created was a Bizarro version of Wonder Woman. Bizarro Wonder Woman, working alongside her Bizarro confederates Batman, Flash, Green Lantern and Hawkgirl, sought to save Bizarro from Bizarro Doomsday by dropping their hyperbolic headquarters on top of him.\n\nAs opposed to her counterpart, Bizarro Wonder Woman uses a lasso that causes those ensnared to tell lies.\ …

https://dc.fandom.com/wiki/Bizarro_Wonder_Woman_(New_Earth)

- Book Title Poetry: April 26,\u00a02012\n\n\nCan't recall where I first saw this but I'll admit it isn't my own original idea. I think it was a link on Twitter or something, which I found whilst poking around. Of course I can't find it again.\n\nBasically, the idea is you grab a few books from a shelf or shelves, stack them up and make a poem from the titles. A simple idea and it can strike gold or come off sounding like a third-grader's attempt at a poetry homework assignment (no offense to third graders).\n\nThe annoying rule is you have to keep the books in the same order you pulled them from the shelves.\n\nLet's try!\n\n\n\n\nKipling's Kim\n\nRobert Levine's Free Ride\n\n […]

https://bluestalkingjournal.com/2012/04/26/book-title-poetry-april-26-2012/

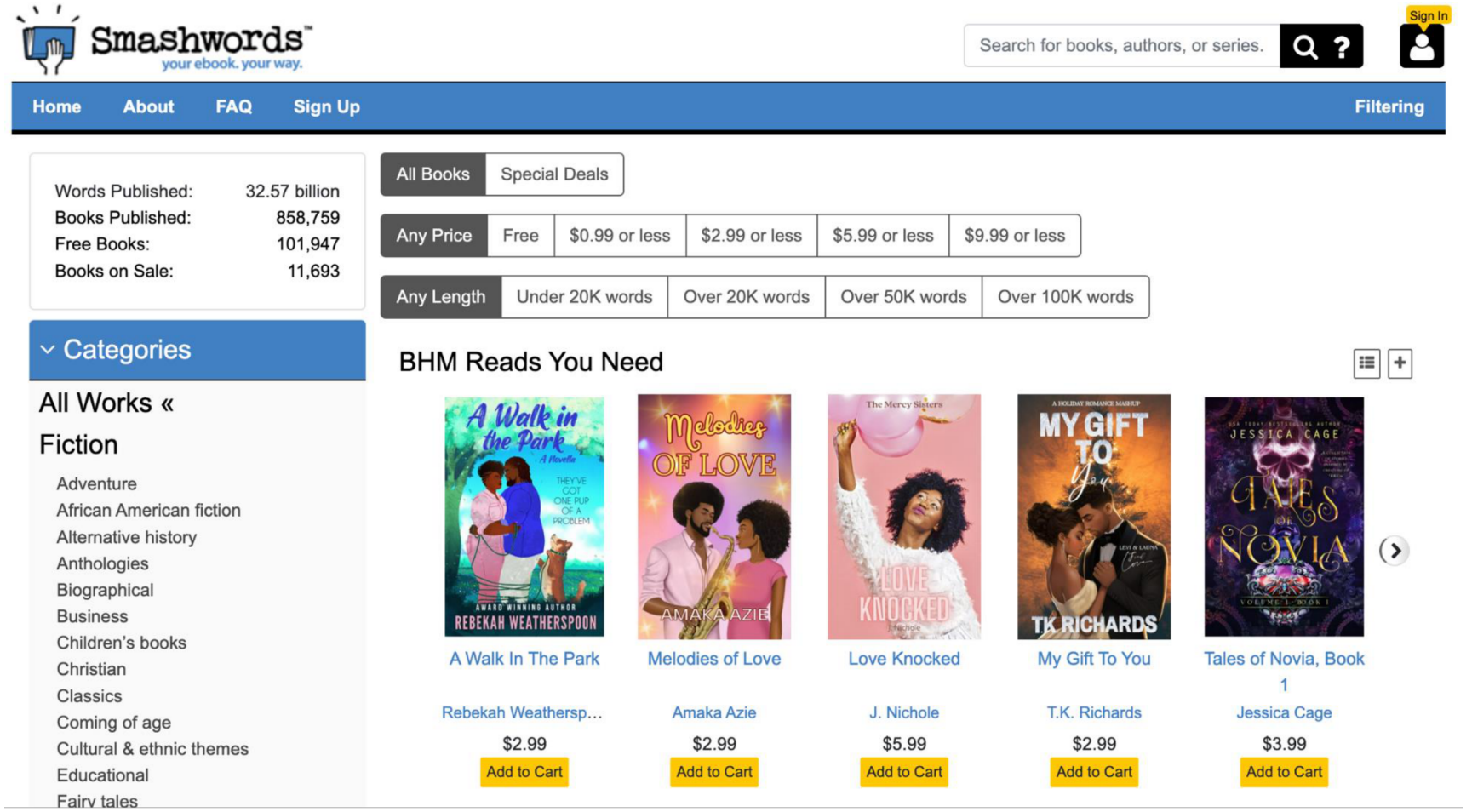
- Hibernate Interview Questions and Answers\n\nHibernate Interview Questions and Answers\n\nHibernate is an open source simple ORM tool. It is a java framework that simplifies the development of java application to interact with the database. Hibernate not only takes care of the mapping from Java classes to database, but also provides data query and retrieval facilities.\n\nWhat is hibernate?\n\nWhat is ORM?\n\nWhat is Hibernate Framework?\n\nWhat is Java Persistence API (JPA)?\n\nWhat are the important benefits of using Hibernate Framework?\n\nWhat are the advantages of Hibernate over JDBC?\n\nWhat is hibernate configuration file?\n\nWhat is hibernate mapping file?\n\ […]

https://www.interviewgig.com/hibernate-interview-questions-and-answers/

- Artificial Intelligence \u2013 should we be worried?\n\nThere\u2019s a lot in the media at the moment concerning Artificial Intelligence, some hailing it as the next industrial revolution, others as Armageddon waiting to happen.\u00a0 I know science fiction over the years has been full of the latter.\u00a0 However, as any writer will tell you a good story needs conflict and in sci-fi what\u2019s better than man vs. machine?.\u00a0 I also know that Stephen Hawking is suggesting we, or at least some of us, need to get of this planet before the end of the century and find a new home before AI becomes too powerful.\u00a0 I just don\u2019t see why it has to be that way.\u00a0 Why does it have to be the alarmist view?\u00a0[…]

https://www.martynfiction.com/life-in-the-future/artificial-intelligence-worried/?replytocom=1403

Bookcorpus.. what's that?



- Scraped ebooks from the internet – highly controversial

Fair use and other concerns

Google swallows 11,000 novels to improve AI's conversation

As writers learn that tech giant has processed their work without permission, the Authors Guild condemns 'blatantly commercial use of expressive authorship'

'It doesn't harm the authors' ... Google's headquarters in Mountain View, California. Photograph: Marcio Jose Sanchez/AP

Arts and Humanities, Law, Regulation, and Policy, Machine Learning

Reexamining "Fair Use" in the Age of AI

Generative AI claims to produce new language and images, but when those ideas are based on copyrighted material, who gets the credit? A new paper from Stanford University looks for answers.

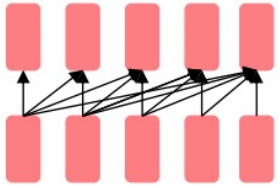
Jun 5, 2023 | Andrew Myers

Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - Very large models and in-context learning

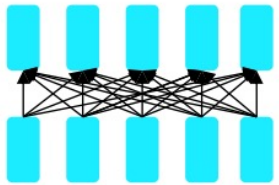
Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.



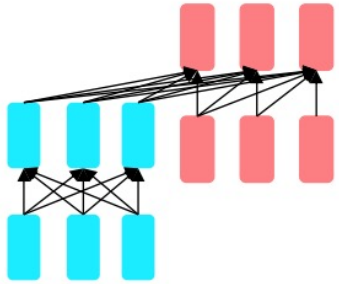
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA, LLaMa



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa

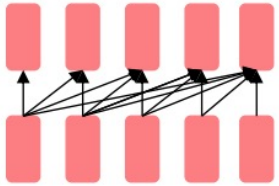


Encoder-Decoders

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

Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.

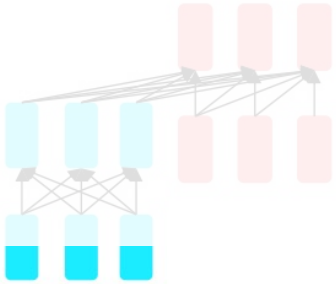


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA

Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa



Encoder-Decoders

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

Pretraining decoders

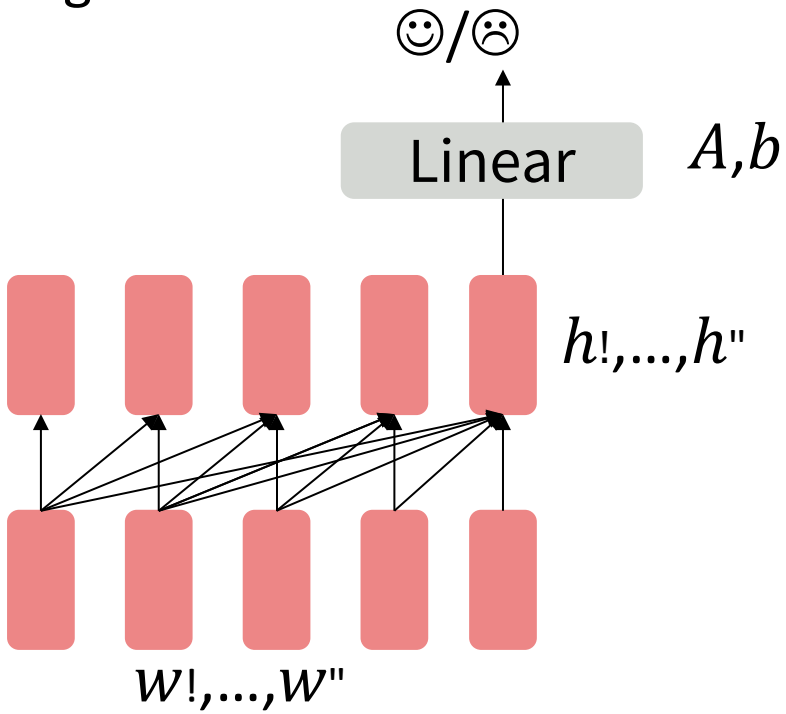
When using language model pretrained decoders, we can ignore that they were trained to model $p_\theta(w_t|w_{1:t-1})$

We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$

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.]

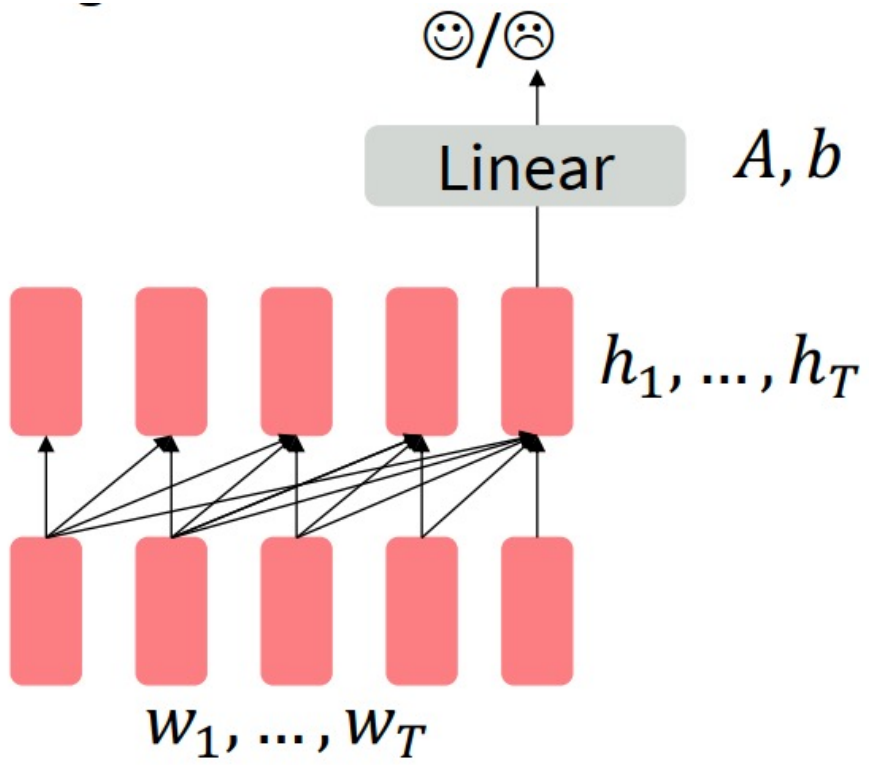
Pretraining decoders

It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p(w_t|w_{1:t-1})$

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)

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



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

Where A,b were pretrained in the language model!

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 BooksCorpus: 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?**

Natural Language Inference: Label pairs of sentences as *entailing/contradictory/neutral*

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door*

} **entailment**

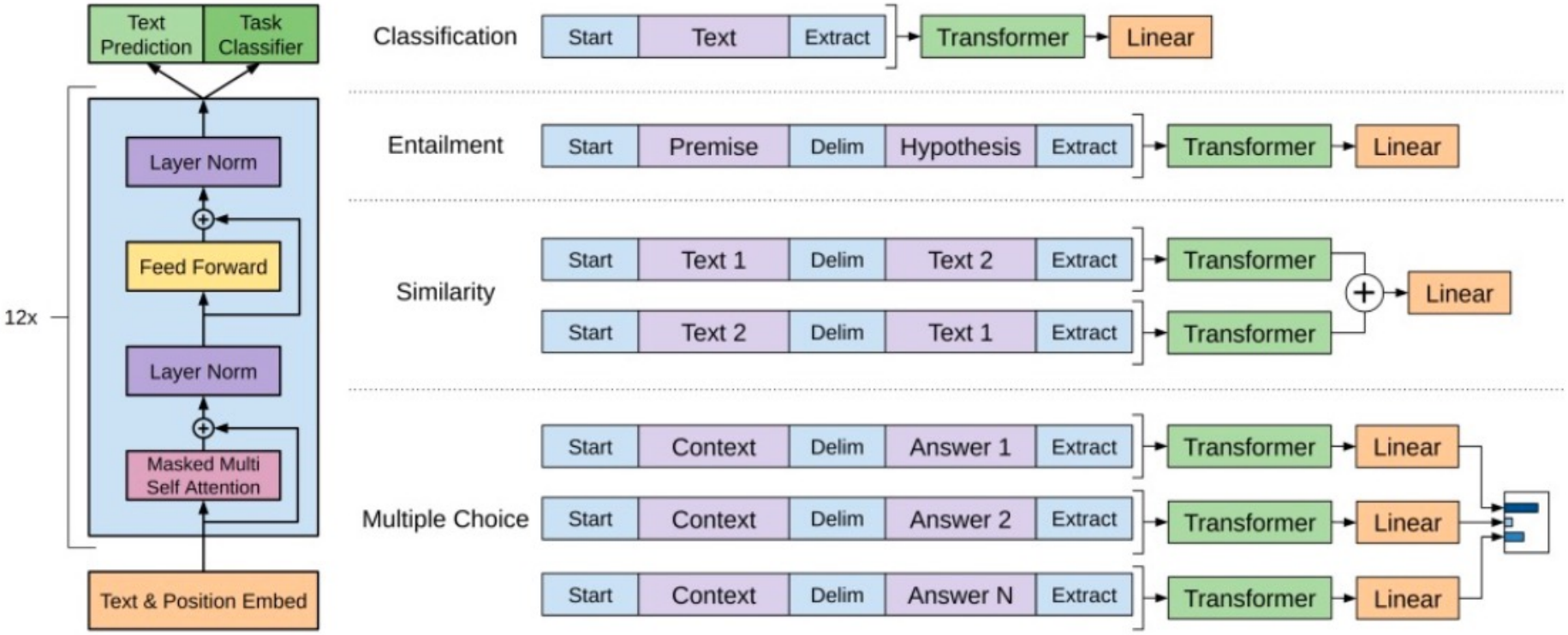
Radford et al., 2018 evaluate on natural language inference.
Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

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

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

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



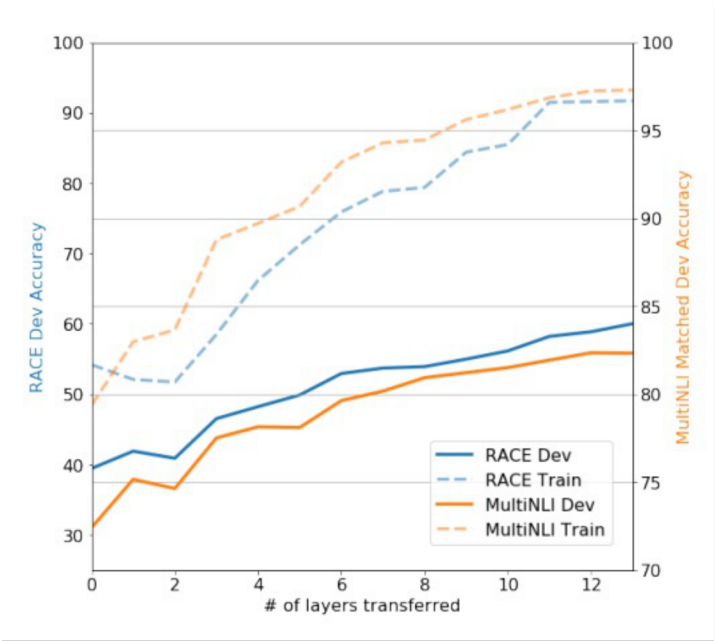
- The linear classifier is applied to the representation of the [EXTRACT] token.

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

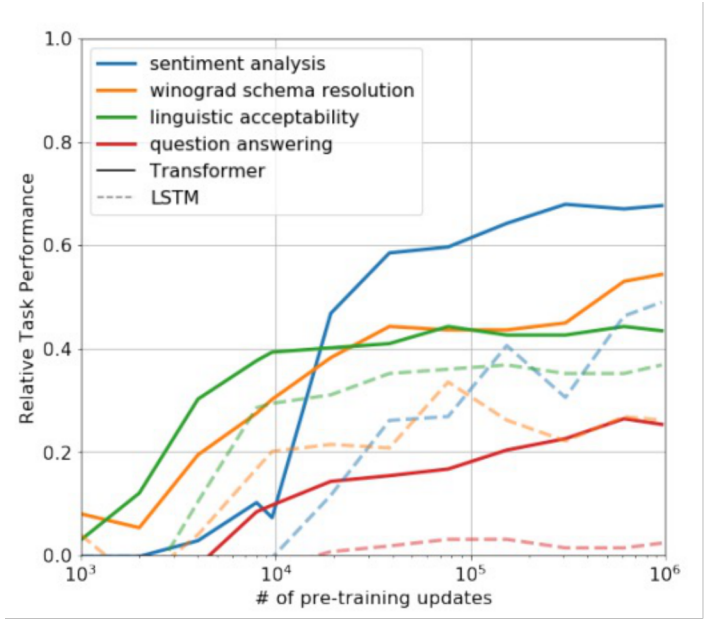
- GPT results on various natural language inference datasets

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

Examining the Effect of Pretraining in GPT [Radford et al., 2018]



As more layers are transferred, performance improves on RACE (a large-scale reading comprehension dataset) and MultiNLI.



Zero-shot performance of Transformer vs. LSTM as a function of the # of pre-training updates.

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

We mentioned how pretrained decoders can be used **in their capacities as language models.**

GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

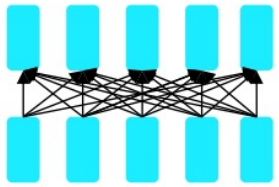
Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after 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.

Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.

Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa

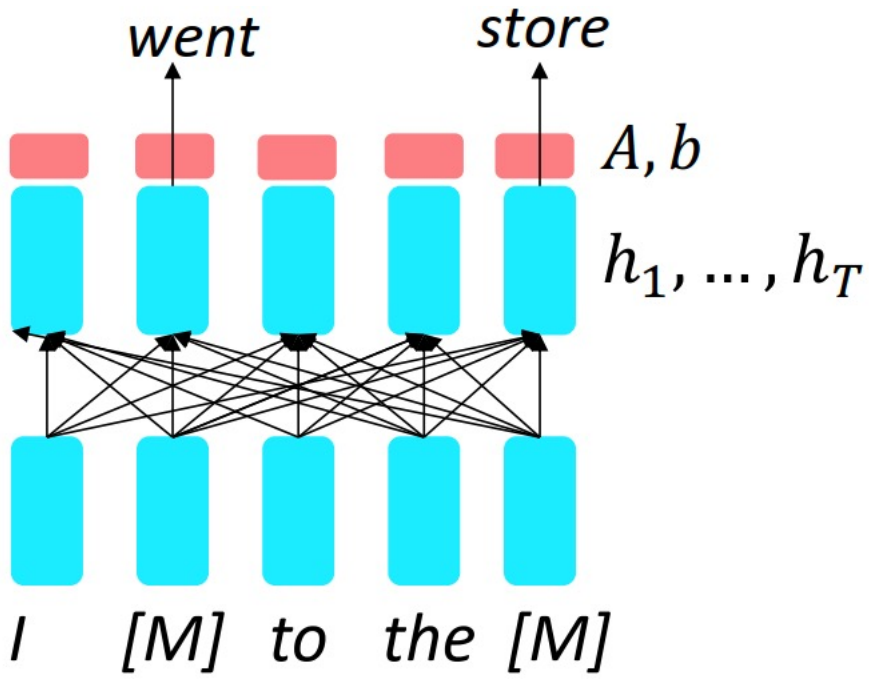
Encoder-Decoders

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

Pretraining encoders: what pretraining objective to use?

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.



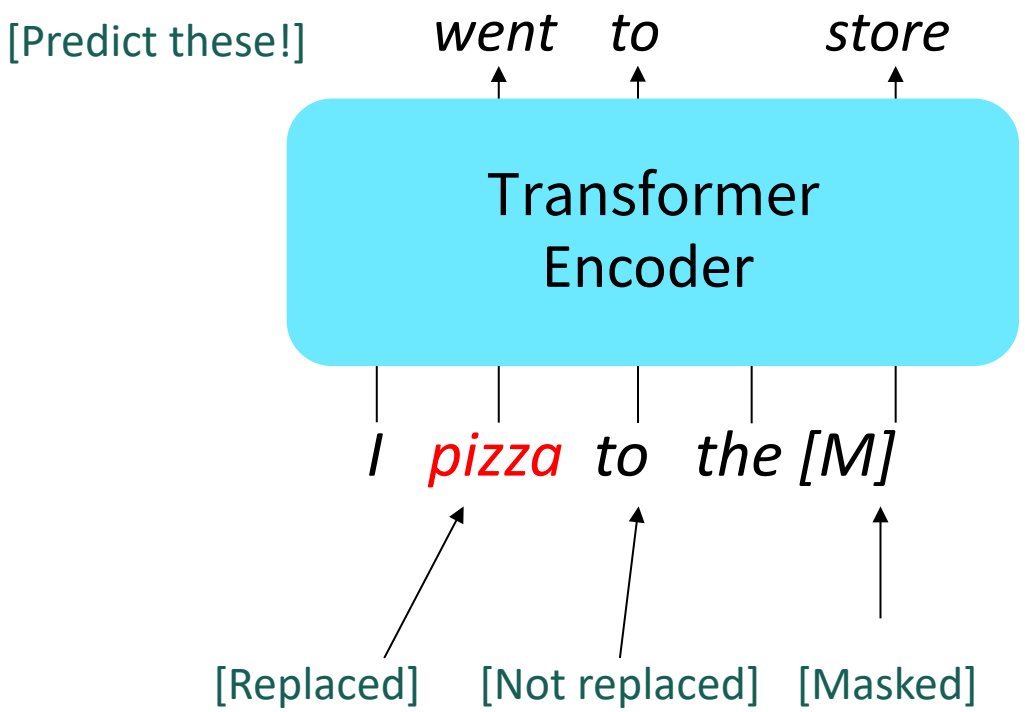
Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$ Called **Masked LM**.

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective, open-sourced their model as the tensor2tensor library, and **released the weights of their pretrained Transformer (BERT)**.

Some more details about Masked LM for BERT:

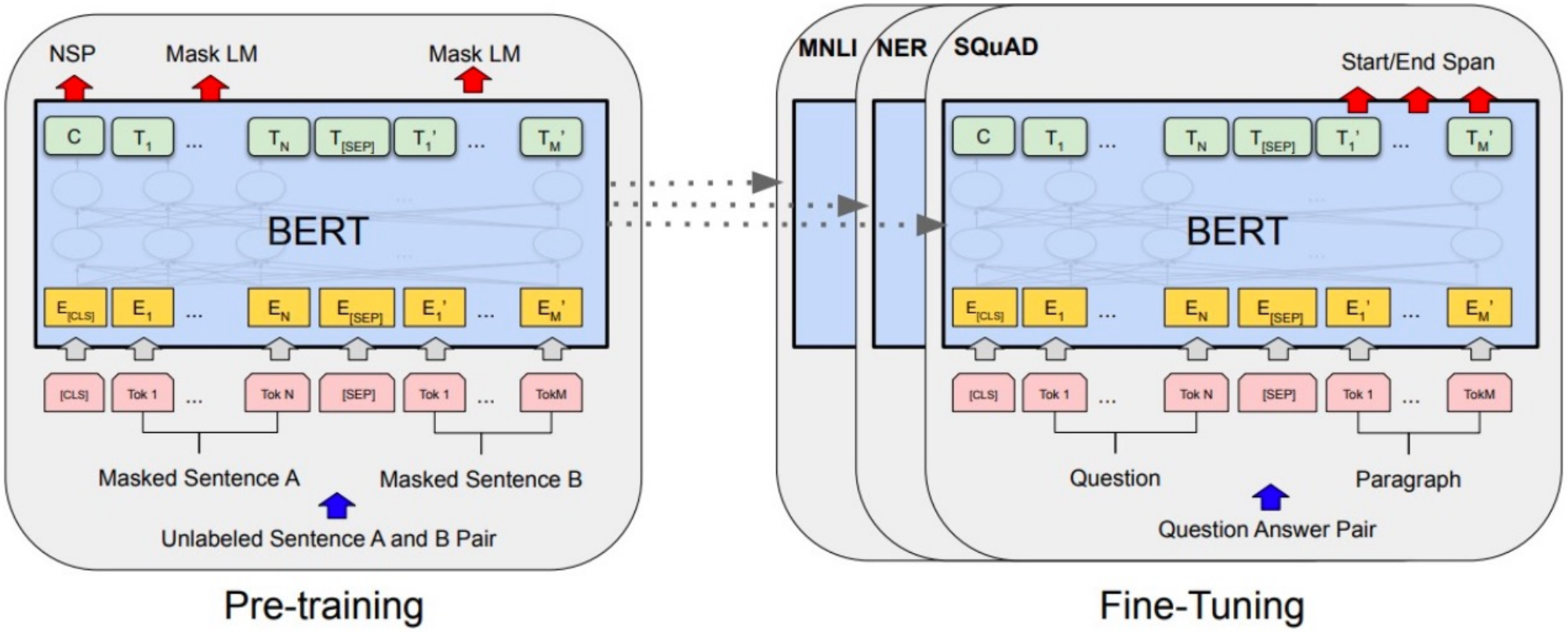
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



[Devlin et al., 2018]

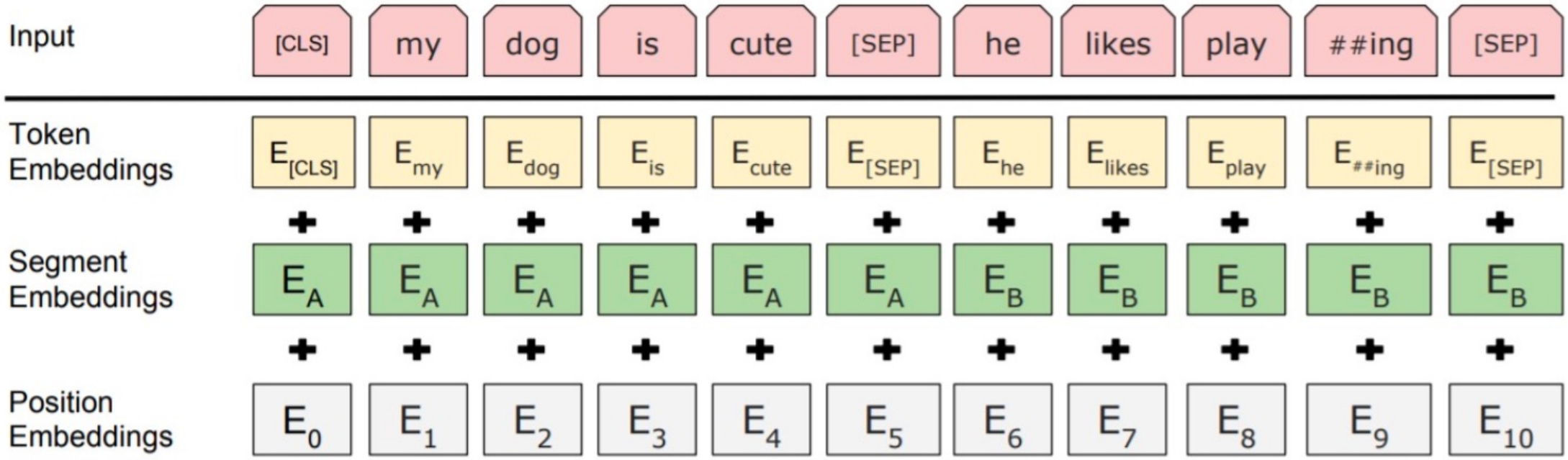
BERT: Bidirectional Encoder Representations from Transformers

- **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 has argued this “next sentence prediction” is not necessary.



BERT: Bidirectional Encoder Representations from Transformers

Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (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)
- Finetuning is practical and common on a single GPU
  - “Pretrain once, finetune many times.”

BERT: Bidirectional Encoder Representations from Transformers

- 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

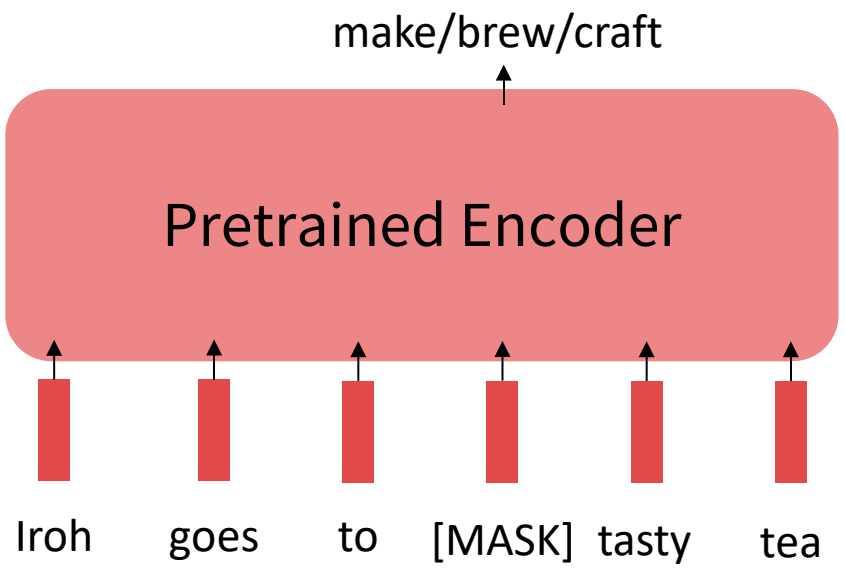
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

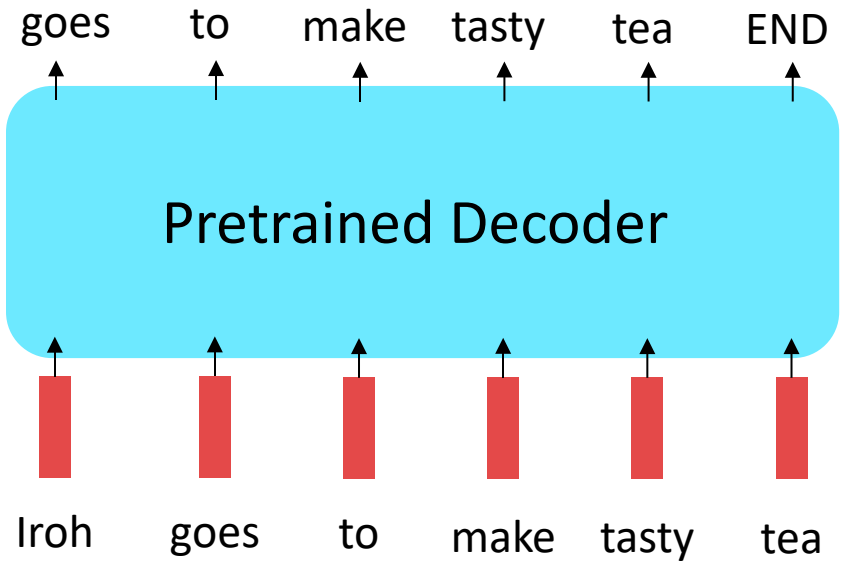
Note that $BERT_{BASE}$ was chosen to have the same number of parameters as OpenAI GPT.

Limitations of pretrained encoders

Those results looked great! Why not used pretrained encoders for everything?

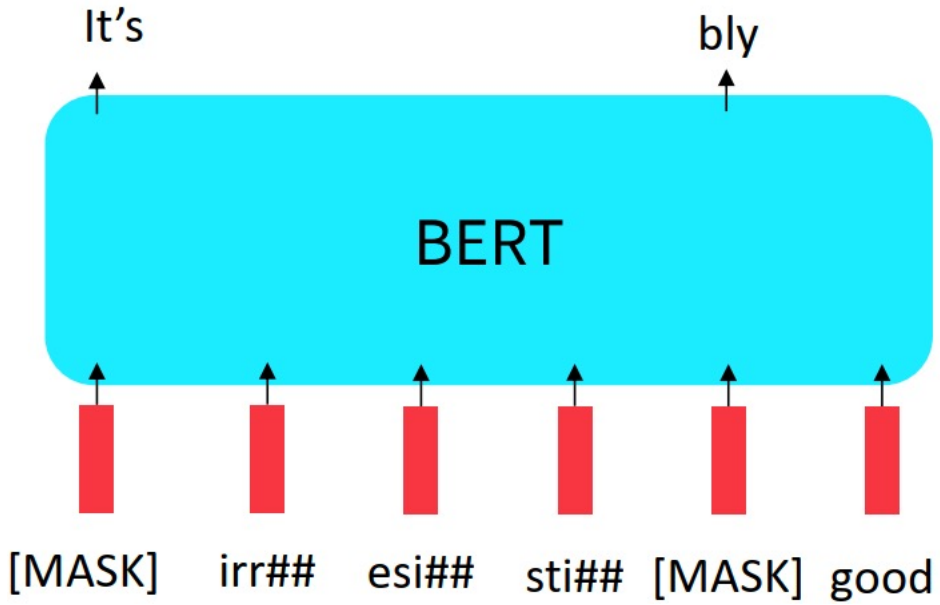
If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.

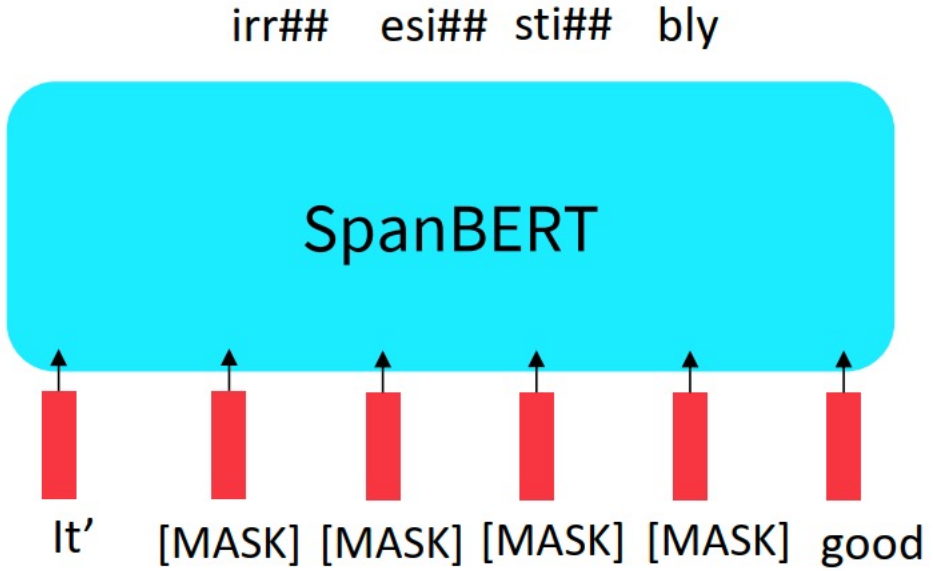




Extensions of BERT

- **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





Extensions of BERT

- A takeaway from the RoBERTa paper: more compute, more data can improve pretraining even when not changing the underlying Transformer encoder.

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Pretraining for three types of architectures

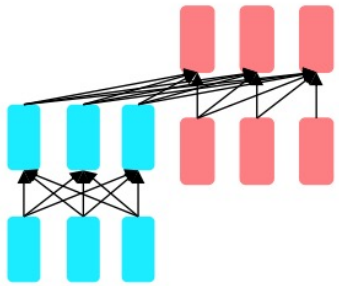
- The neural architecture influences the type of pretraining, and natural use cases.

Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA

Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa



Encoder-Decoders

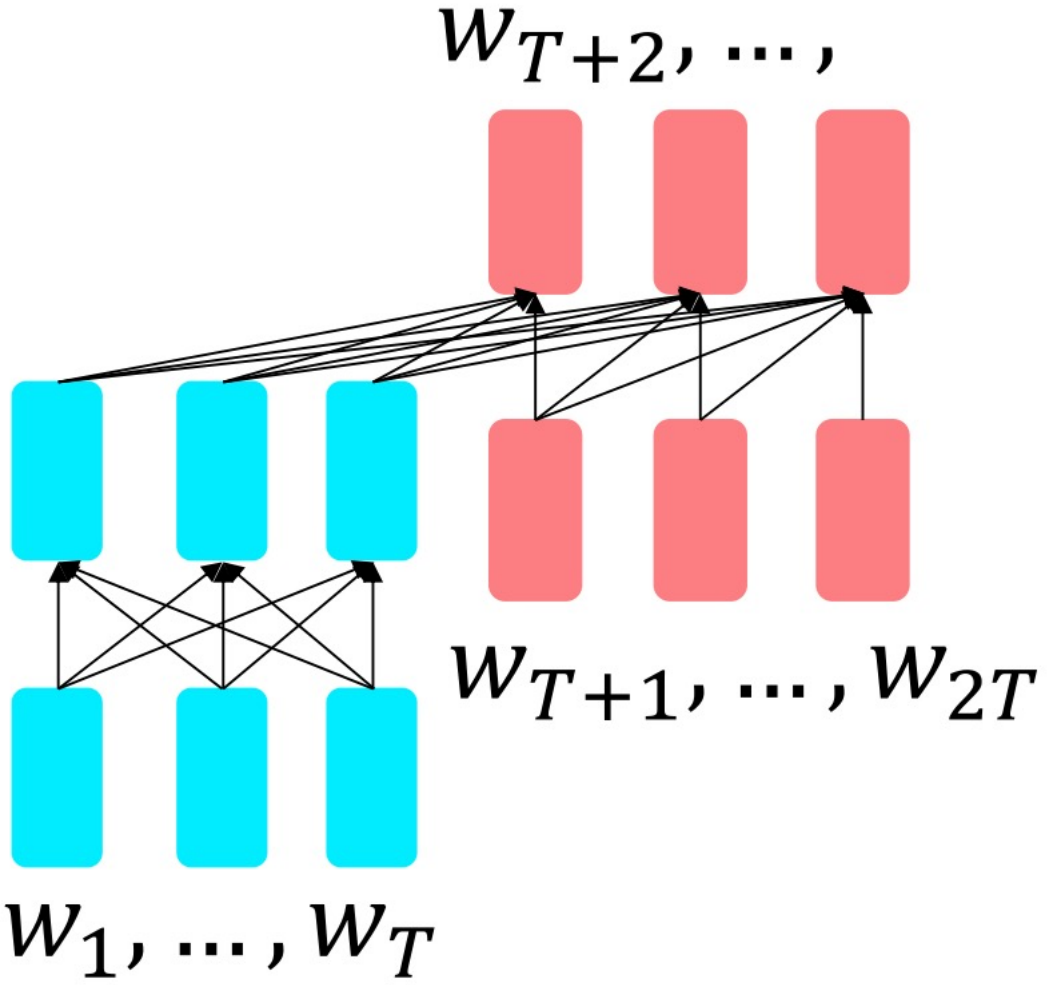
- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- Examples: Transformer, T5, Meena

Pretraining encoder-decoders: what pretraining objective to use?

For **encoder-decoders**, we could do something like **language modeling**, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$

The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.



Pretraining encoder-decoders: what pretraining objective to use?

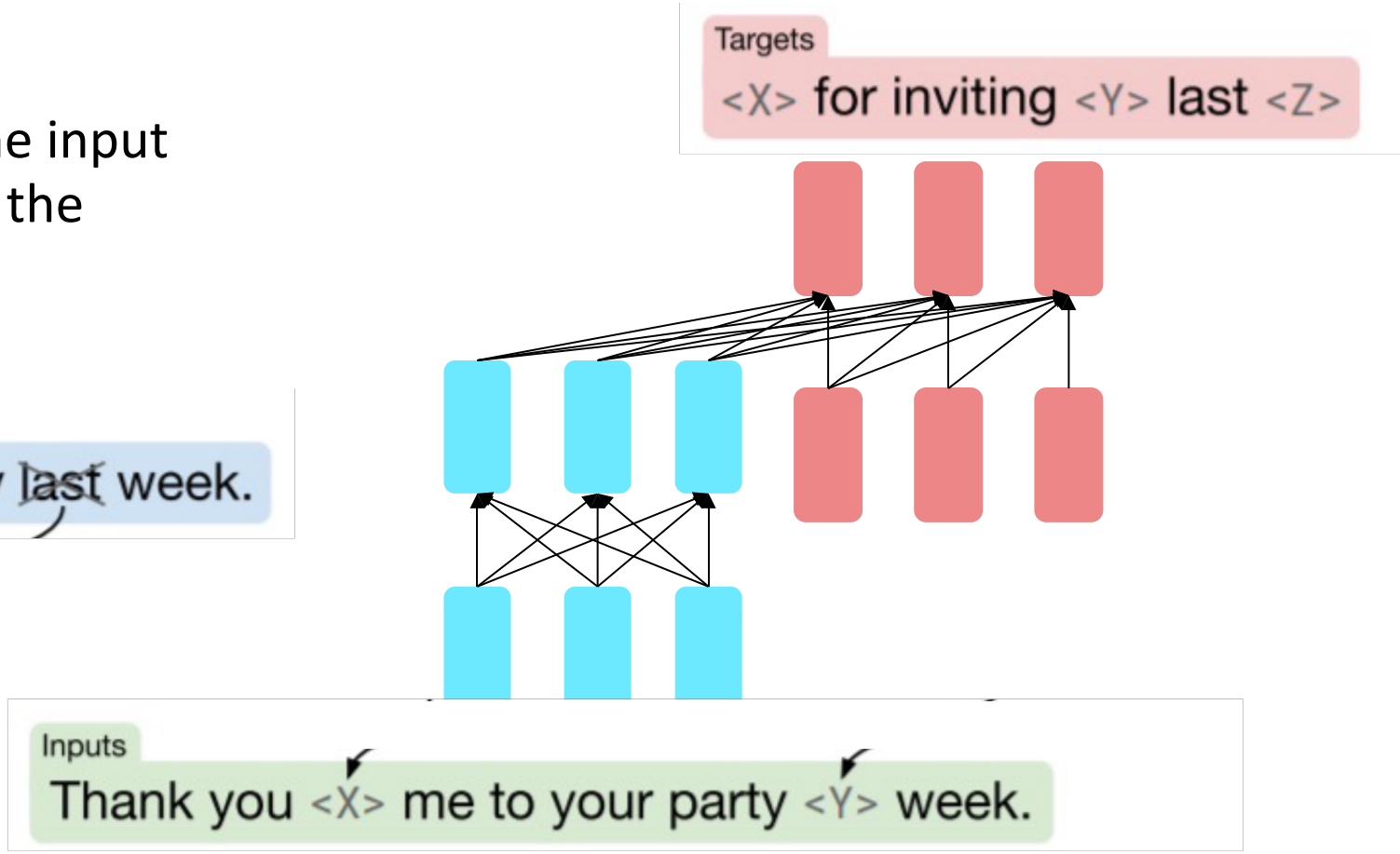
What Raffel et al., 2018 found to work best was **span corruption.** Their model: **T5**.

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

Thank you for inviting me to your party last week.

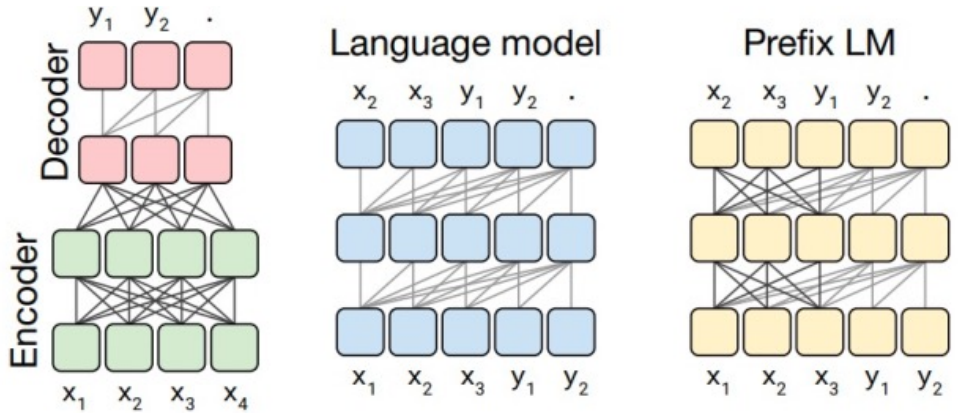
This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



[Raffel et al., 2018]

Pretraining encoder-decoders: what pretraining objective to use?

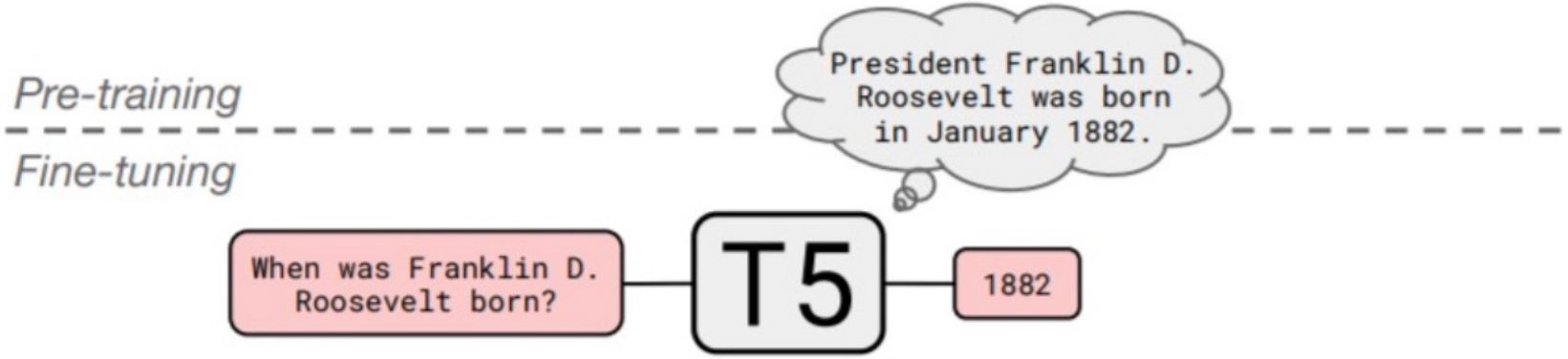
- Raffel et al., 2018 found that
  - encoder-decoders work better than decoders for their tasks
  - Span corruption (denoising) work better than simply language modeling.



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

Pretraining encoder-decoders: what pretraining objective to use?

- A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.



- NQ: Natural Questions
- WQ: WebQuestions
- TQA: Trivia QA
- All "open-domain" versions

	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**	

Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - Very large models and in-context learning

GPT-3, In-context learning, and very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- 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.

GPT3 pretraining data

Dataset	Tokens (billion)	Assumptions	Tokens per byte (Tokens / bytes)	Ratio	Size (GB)
Web data	**410B**	–	*0.71*	*1:1.9*	**570**
WebText2	**19B**	*25% > WebText*	*0.38*	*1:2.6*	*50*
Books1	**12B**	*Gutenberg*	*0.57*	*1:1.75*	*21*
Books2	**55B**	*Bibliotik*	*0.54*	*1:1.84*	*101*
Wikipedia	**3B**	*See RoBERTa*	*0.26*	*1:3.8*	*11.4*
Total	**499B**			*753.4GB*	

Table. GPT-3 Datasets. Disclosed in **bold**. Determined in *italics*.

Alan D. Thompson, GPT-3.5 + ChatGPT: An illustrated overview, https://lifearchitect.ai/chatgpt/

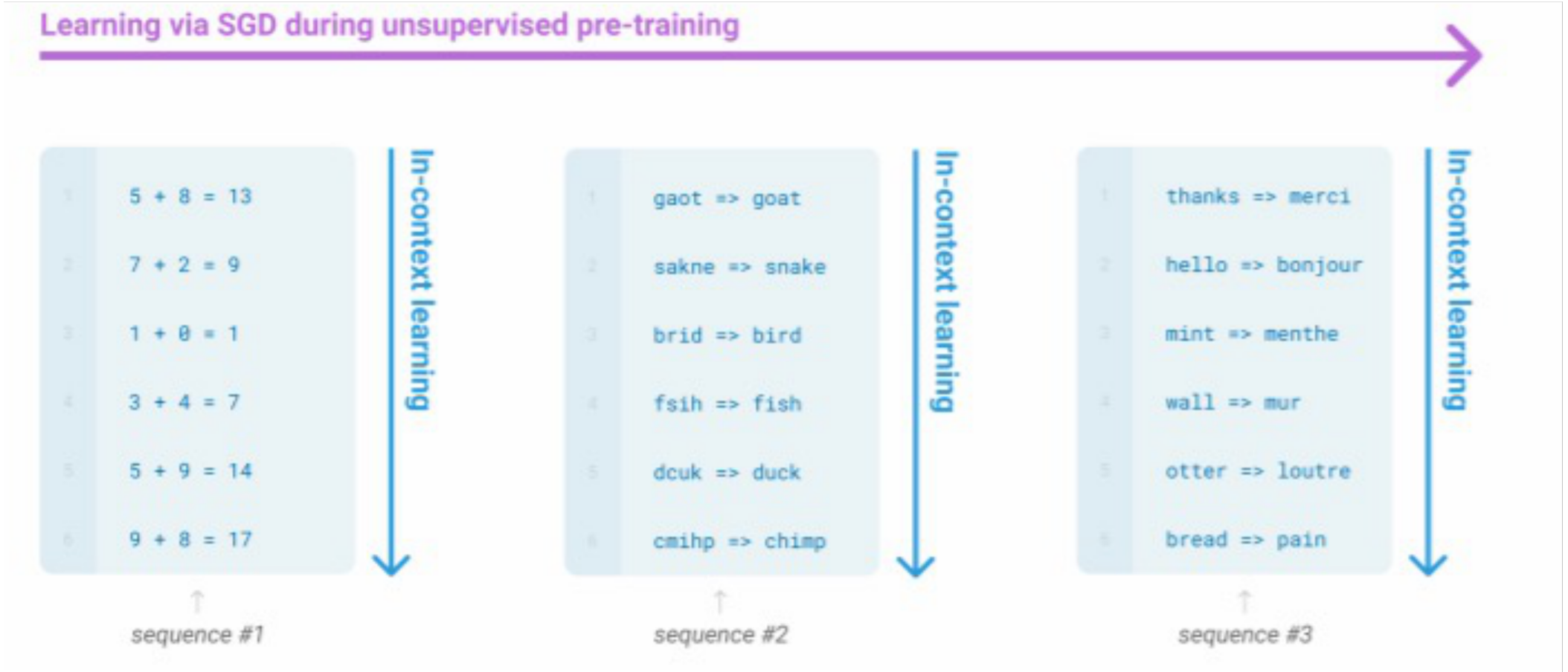
GPT-3, In-context learning, and very large models

- The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing

- Input (prefix within a single Transformer decoder context):
  - “ thanks -> merci
  - hello -> bonjour
  - mint -> menthe
  - otter -> ”
- Output (conditional generations):
  - loutre…”

GPT-3, In-context learning, and very large models

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



GPT-3, In-context learning, and very large models

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

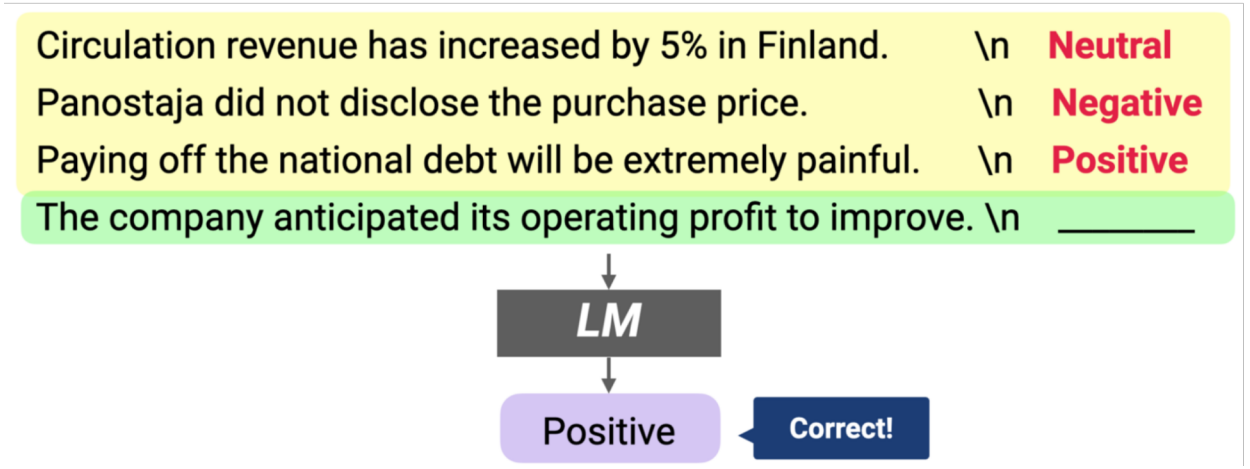
Few-shot

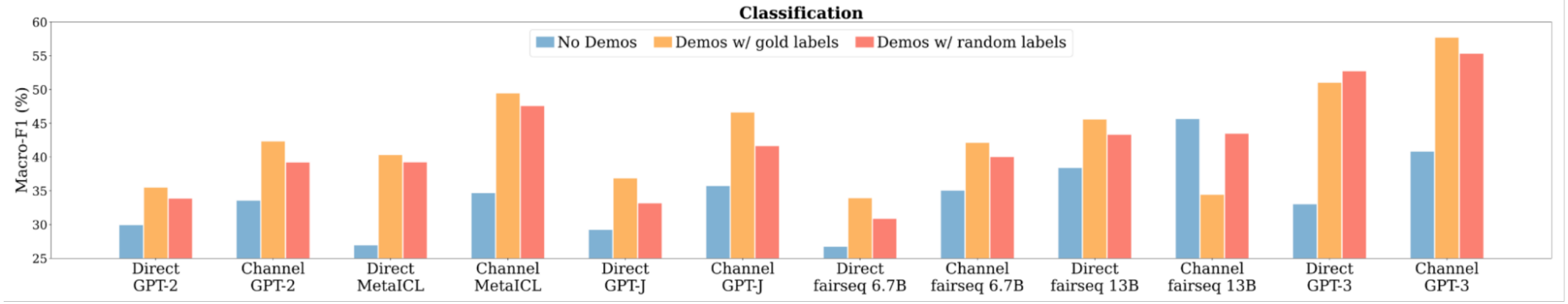
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>                           ← prompt
```

How do LLMs generalize?

- In-context learning is remarkable, but is it *learning*?

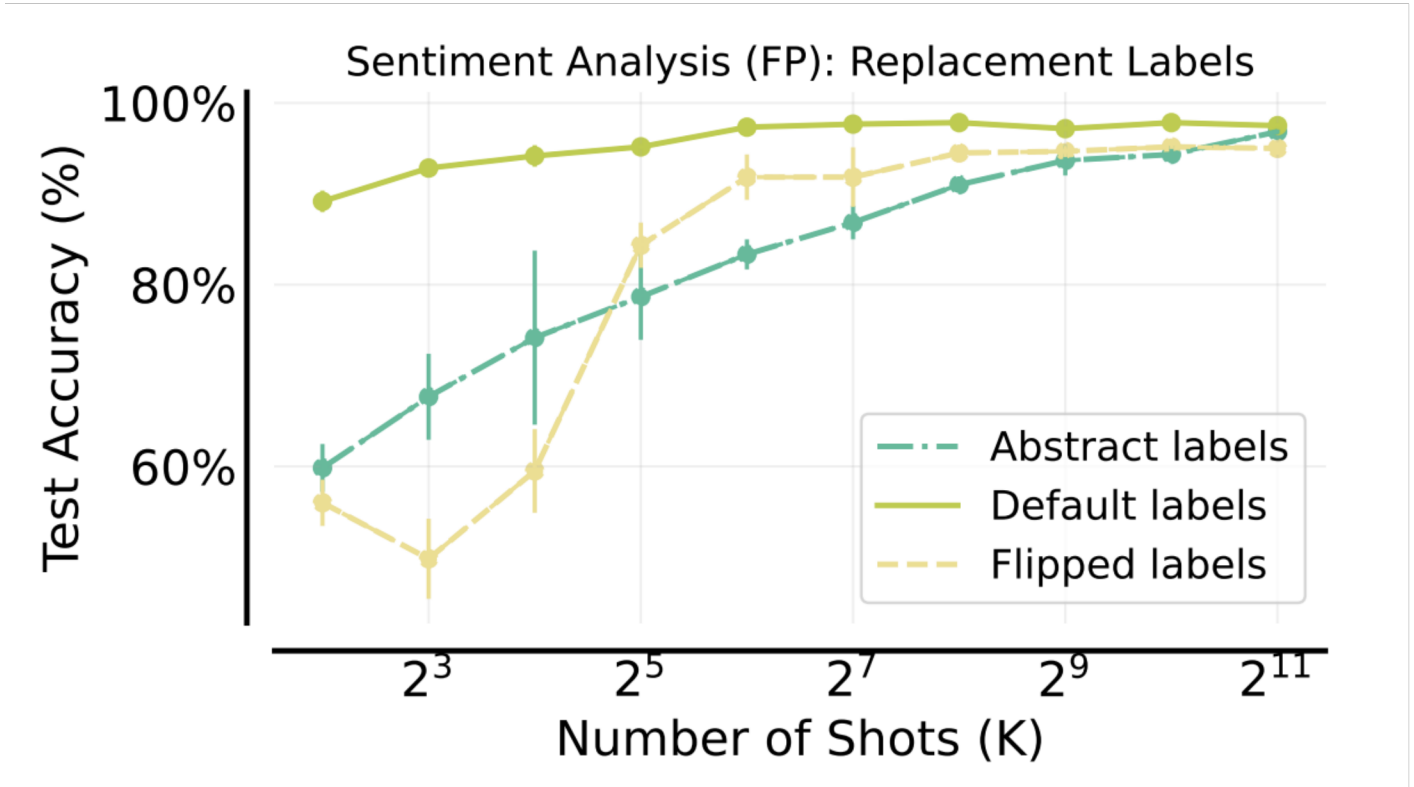




Models can do well on learning problems .. with random labels!

How do LLMs generalize

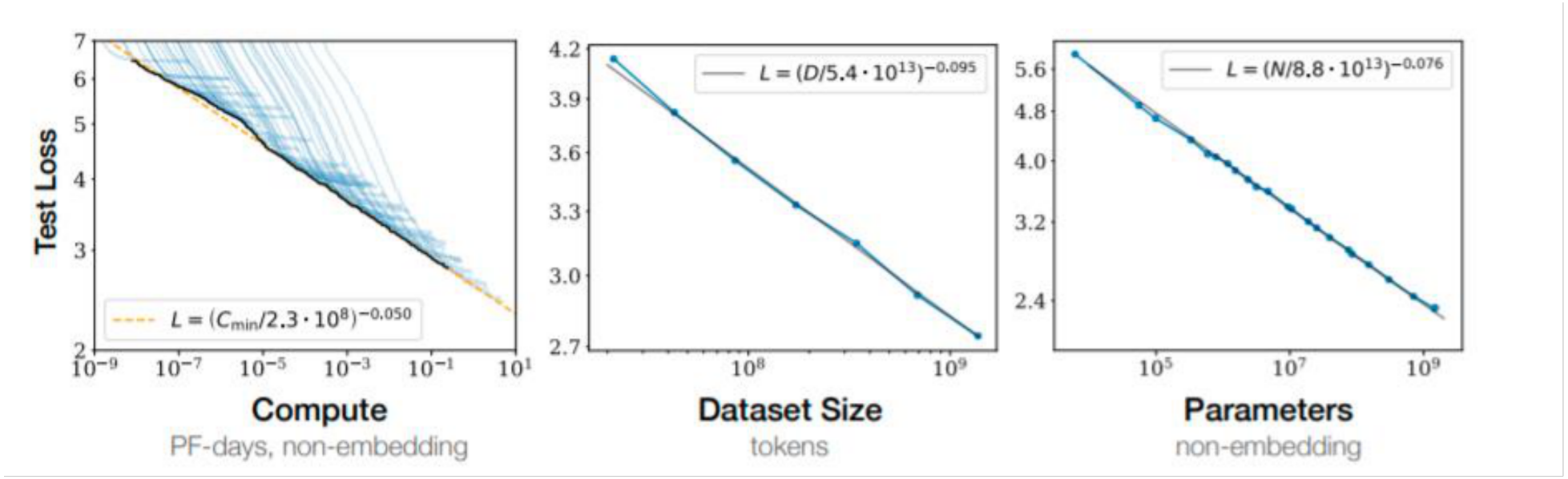
- LLM behavior is often complex, a mix of many different things happening



With big models (Gemini), you get a combination of 'infer the task' and 'learn the task'

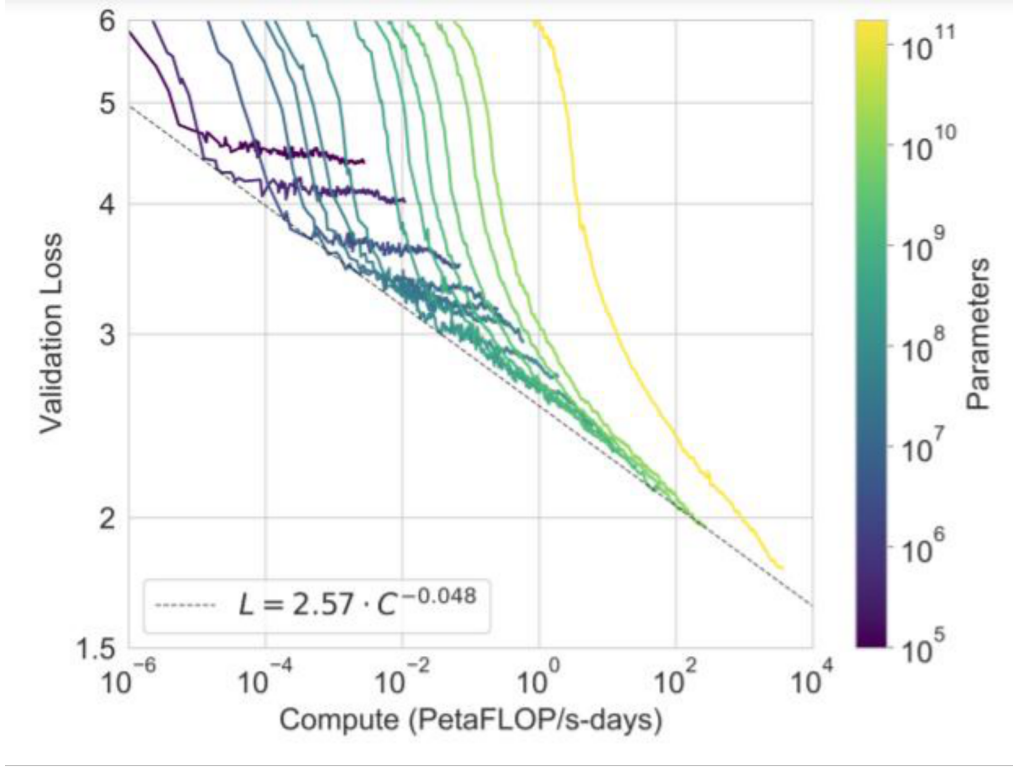
Explanations (and mechanisms) underlying LM behavior are complex!

Why scale? Scaling laws



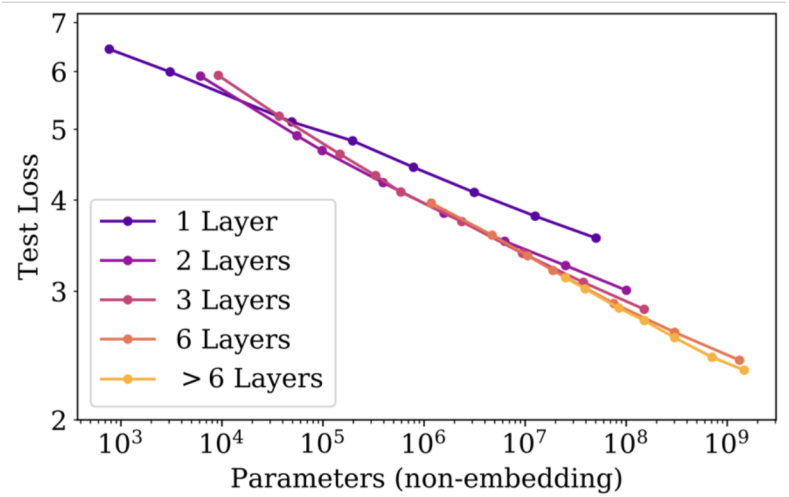
- Empirical observation: scaling up models leads to reliable gains in perplexity

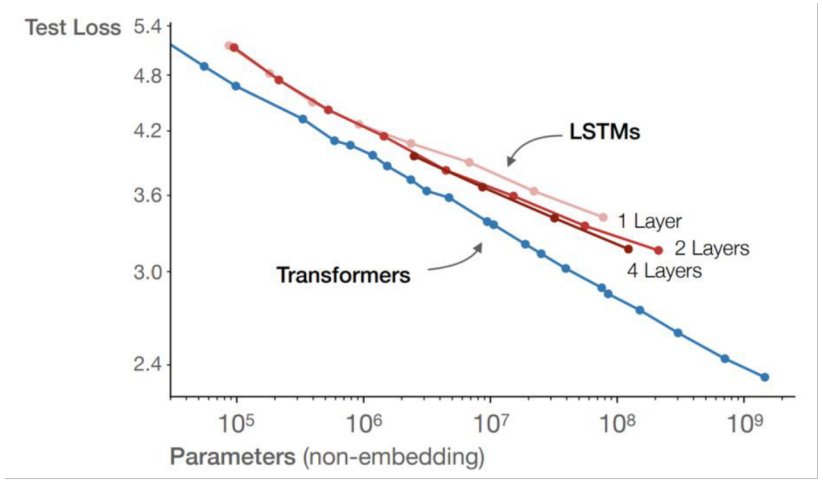
Scaling can help identify model size – data tradeoffs

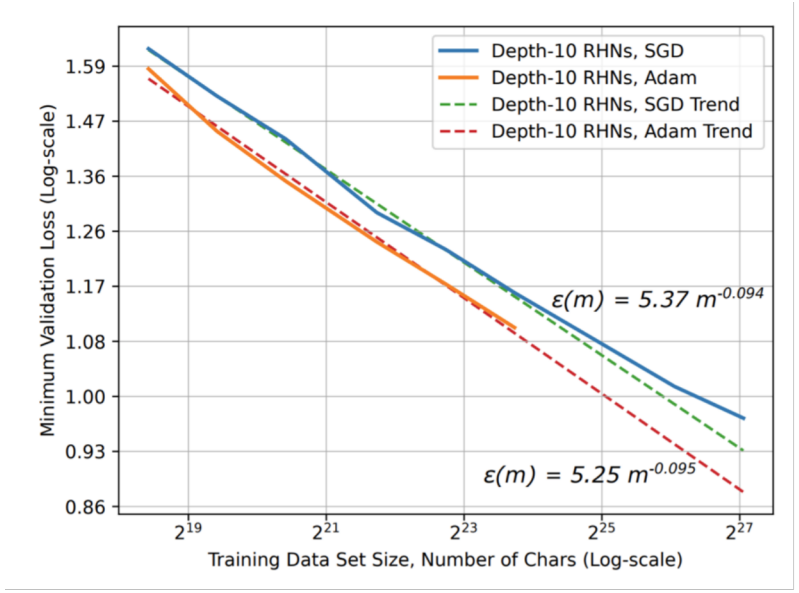


- Modern observation: train a big model that's not fully converged.

Scaling laws for many other interesting architecture decisions







- Predictable scaling helps us make intelligent decisions about architectures etc.

Scaling Efficiency: how do we best use our compute

GPT-3 was **175B parameters** and trained on **300B** tokens of text.

Roughly, the cost of training a large transformer scales as **parameters*tokens**

Did OpenAI strike the right parameter-token data to get the best model? No.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

This 70B parameter model is better than the much larger other models!

Thank you