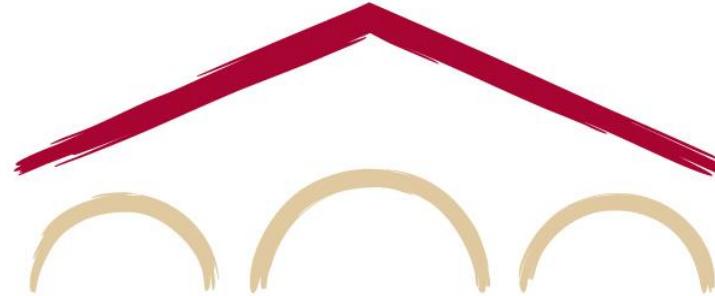


# Natural Language Processing with Deep Learning

**CS224N/Ling284**



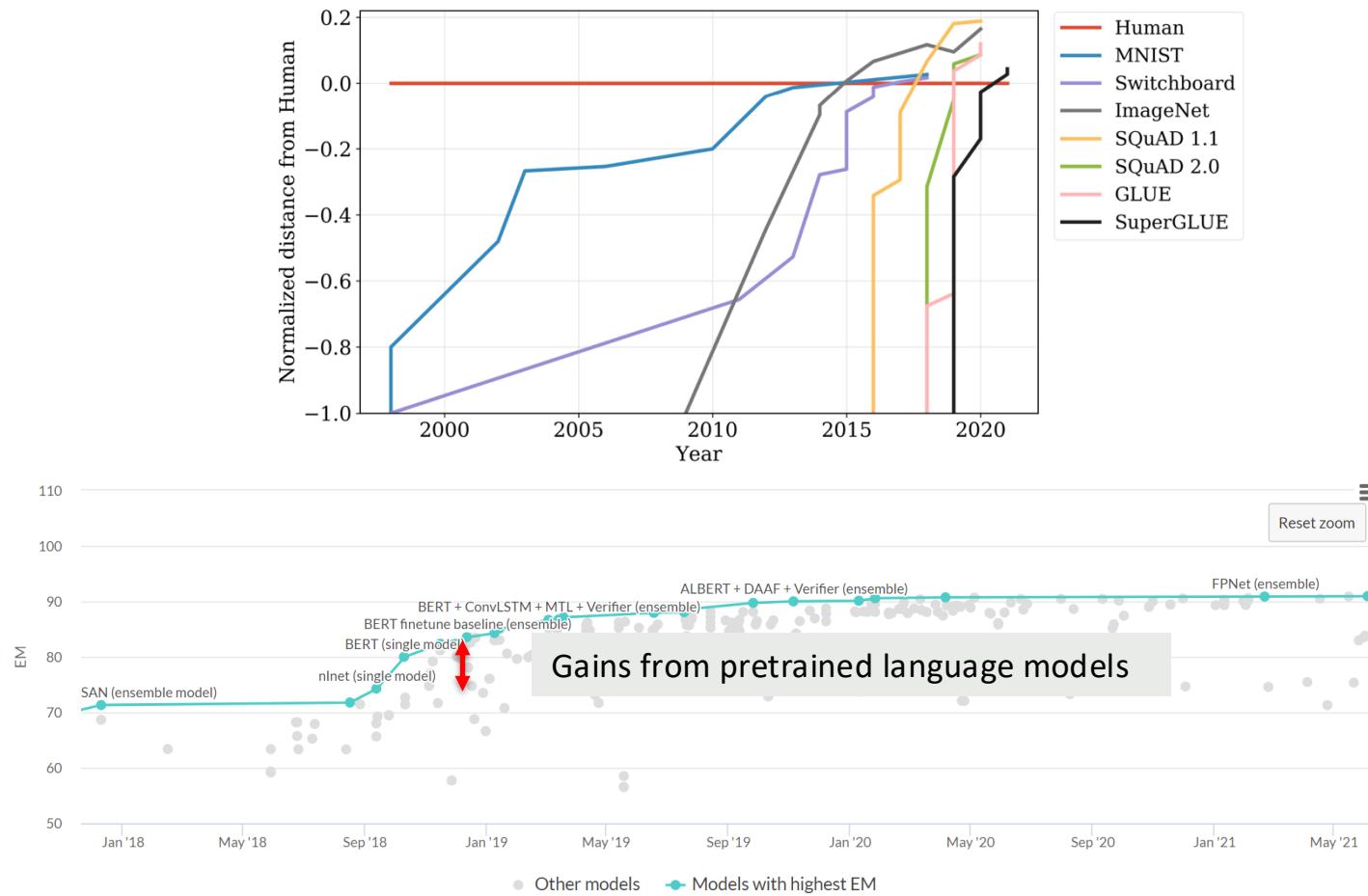
Diyi Yang

Lecture 6: Pretraining

# Lecture Plan

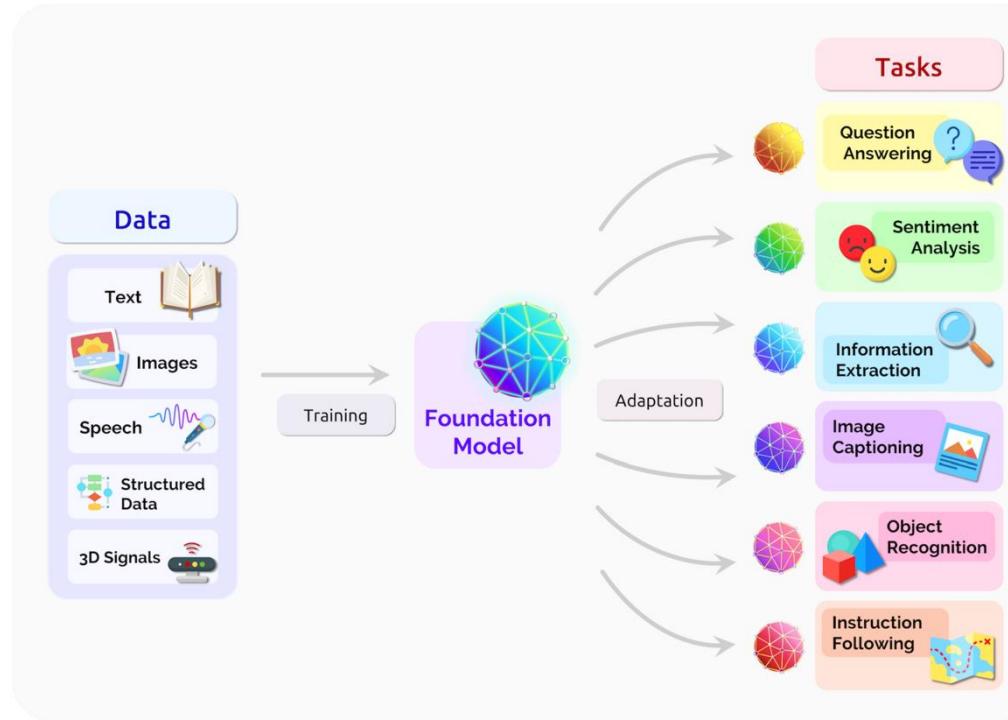
1. Pretraining – motivation (10 mins)
2. Subword modeling (10 mins)
3. Motivating model pretraining from word embeddings (10 mins)
4. Model pretraining three ways (25 mins)
  - Decoders
  - Encoders
  - Encoder-Decoders
5. Interlude: what do we think pretraining is teaching? (10 mins)
6. Very large models and in-context learning (10 mins)

# The pretraining revolution



Pretraining has had a major, tangible impact on how well NLP systems work

# 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

# 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		vocab mapping	embedding
Common words	hat	→	pizza (index)	
	learn	→	tasty (index)	
Variations	taaaaasty	→	UNK (index)	
	laern	→	UNK (index)	
misspellings				
novel items	Transformerify	→	UNK (index)	

# 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)**.
- 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 subword vocabulary.

1. Start with a vocabulary containing only characters and an “end-of-word” symbol.
2. Using a corpus of text, find the most common adjacent characters “a,b”; add “ab” as a subword.
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.

# 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	  
	laern	→ la## ern##	 
misspellings	Transformerify	Transformer## ify	 
novel items			

# Outline

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
  1. Encoders
  2. Encoder-Decoders
  3. Decoders
4. What do we think pretraining is teaching?

# 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.

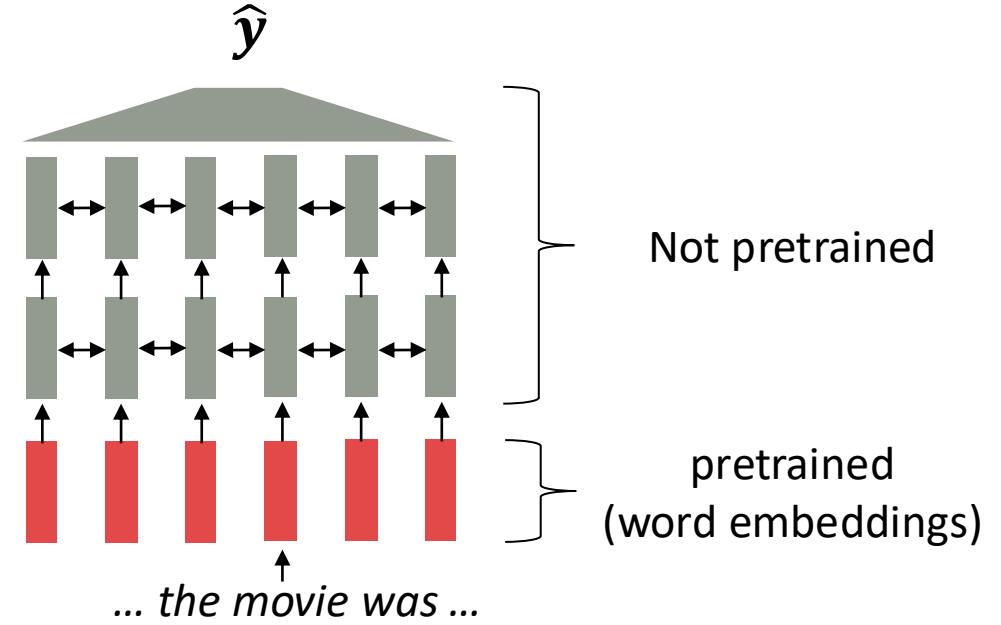
# Where we were: pretrained word embeddings

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!

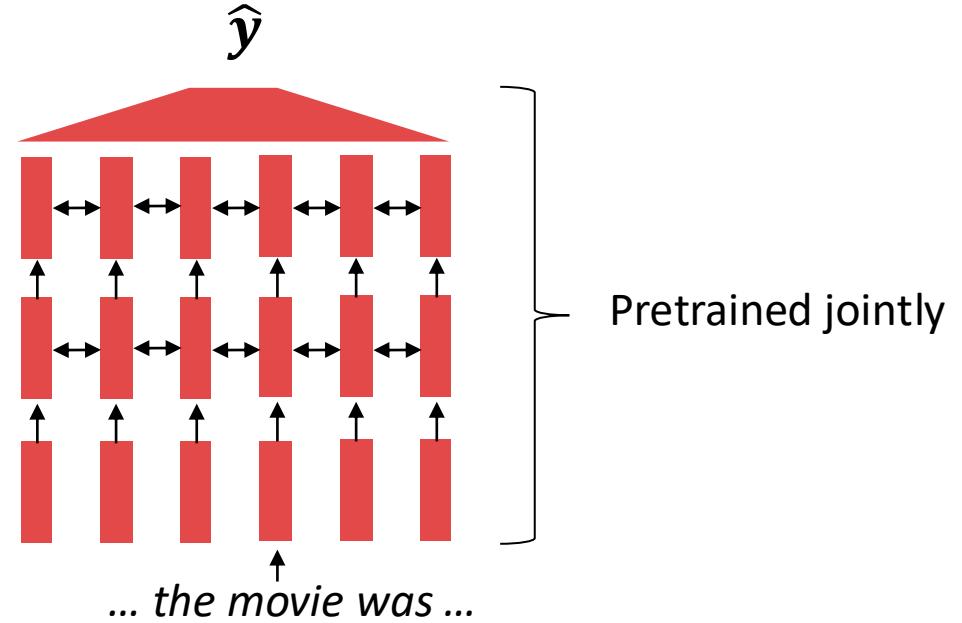


[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

# 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 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?

Stanford University is located in \_\_\_\_\_, California.

# What can we learn from reconstructing the input?

I put \_\_\_ fork down on the table.

# What can we learn from reconstructing the input?

The woman walked across the street,  
checking for traffic over \_\_\_\_ shoulder.

# What can we learn from reconstructing the input?

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

# 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 \_\_\_\_.

# What can we learn from reconstructing the input?

I was thinking about the sequence that goes

1, 1, 2, 3, 5, 8, 13, 21, \_\_\_\_\_

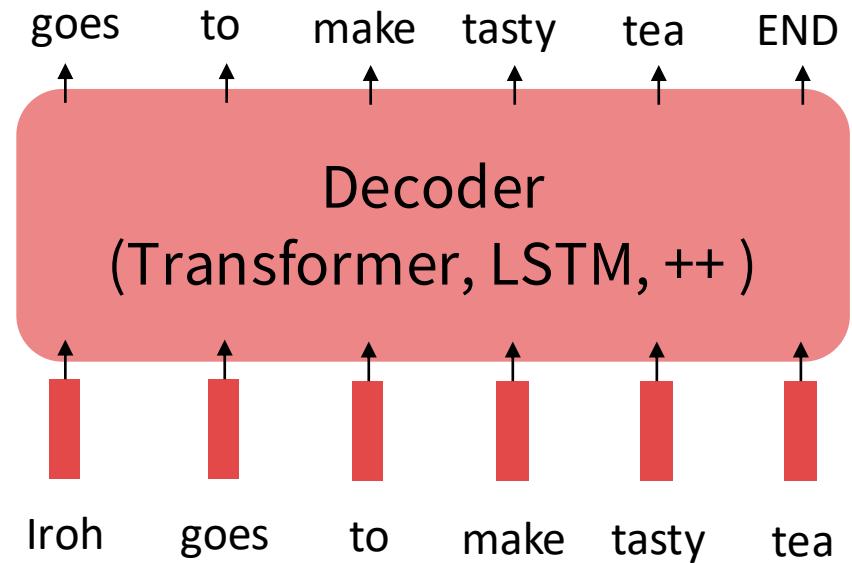
# Pretraining through language modeling [Dai and Le, 2015]

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.

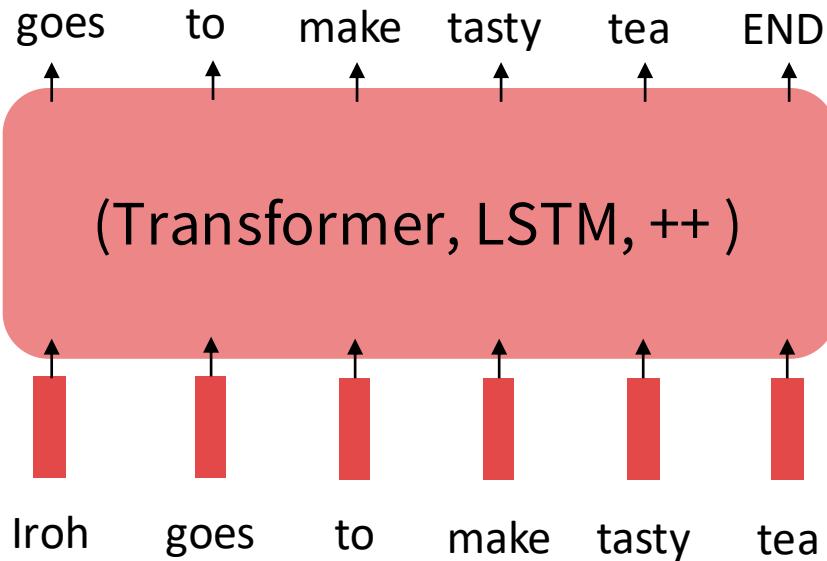


# The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

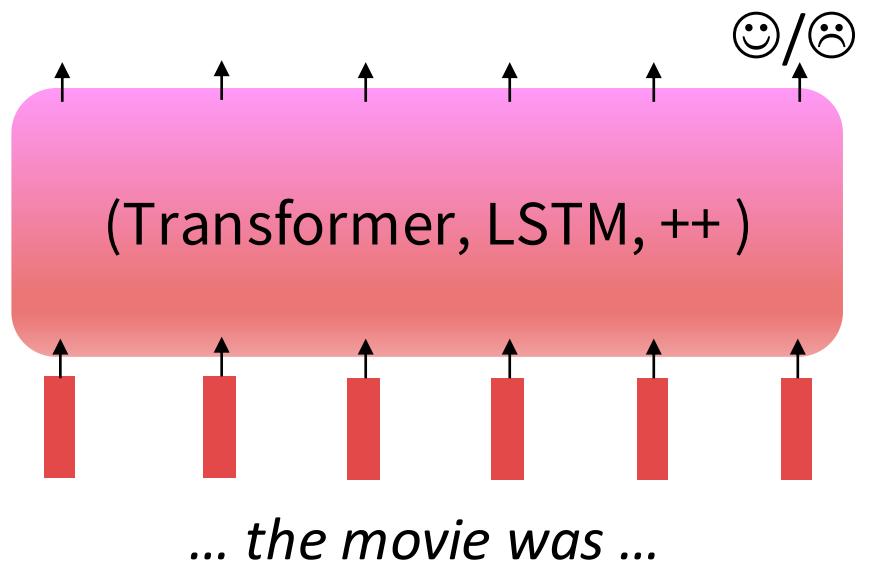
## Step 1: Pretrain (on language modeling)

Lots of text; learn general things!

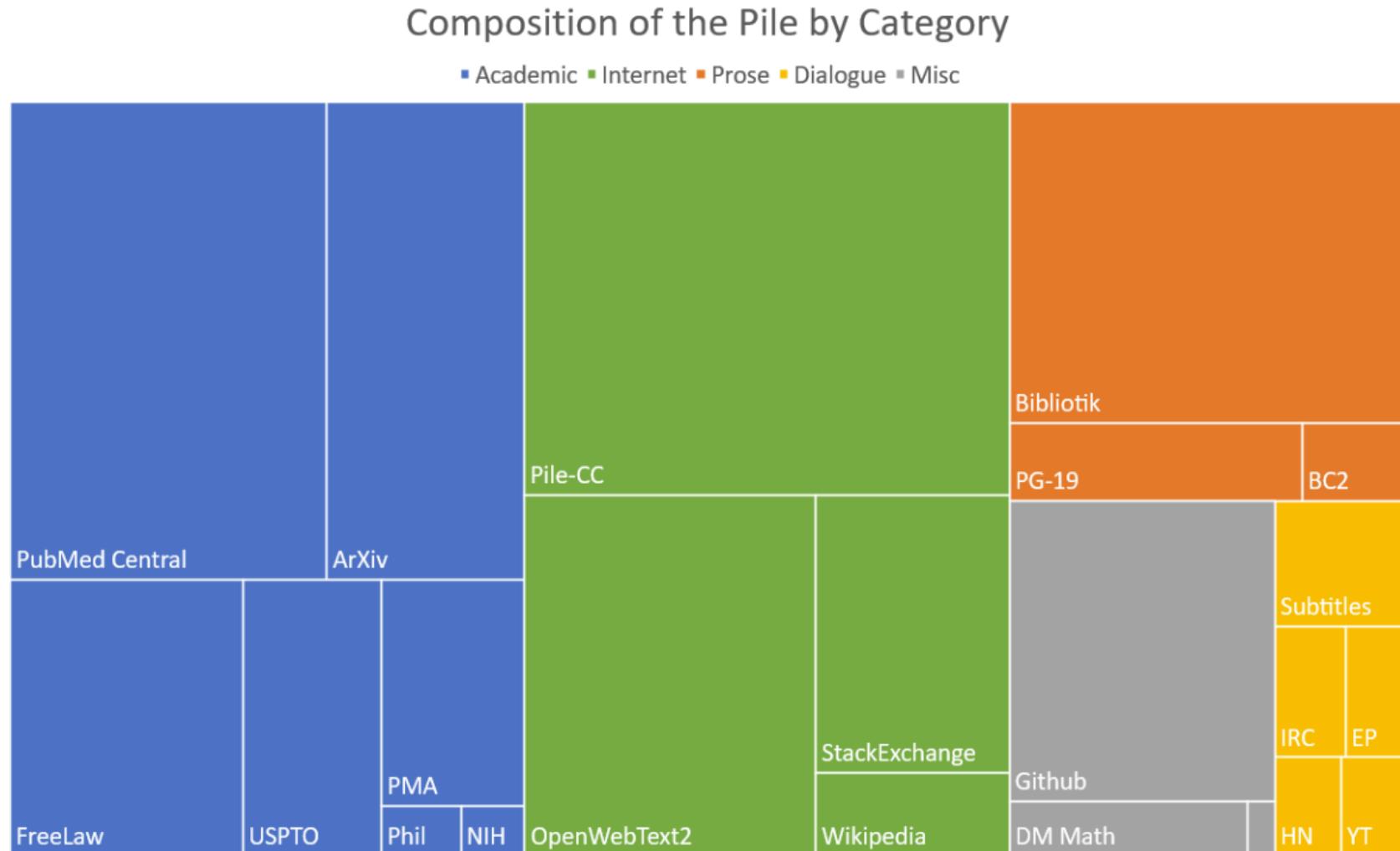


## Step 2: Finetune (on your task)

Not many labels; adapt to the task!



# Where does this data come from?



Model	Training Data
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed

# Bookcorpus.. what's that?

 **Smashwords™**  
your ebook. your way.

Search for books, authors, or series.    

Home   About   FAQ   Sign Up   Filtering

Words Published: 32.57 billion  
Books Published: 858,759  
Free Books: 101,947  
Books on Sale: 11,693

All Books   Special Deals

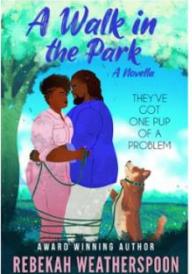
Any Price   Free   \$0.99 or less   \$2.99 or less   \$5.99 or less   \$9.99 or less

Any Length   Under 20K words   Over 20K words   Over 50K words   Over 100K words

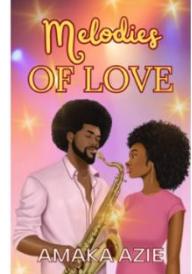
Categories

All Works «  
Fiction  
Adventure  
African American fiction  
Alternative history  
Anthologies  
Biographical  
Business  
Children's books  
Christian  
Classics  
Coming of age  
Cultural & ethnic themes  
Educational  
Fairy tales

BHM Reads You Need



A Walk In The Park  
REBEKAH WEATHERSPOON



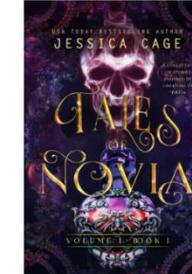
Melodies of Love  
AMAKA AZIE



The Mercy Sisters  
Love Knocked  
J. Nichole



A HOLIDAY ROMANCE MASHUP  
MY GIFT TO YOU  
T.K. RICHARDS



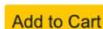
USA TODAY BESTSELLING AUTHOR  
JESSICA CAGE  
TALES OF NOVIA, BOOK 1  
1

A Walk In The Park   Melodies of Love   Love Knocked   My Gift To You   Tales of Novia, Book 1

Rebekah Weathersp...   Amaka Azie   J. Nichole   T.K. Richards   Jessica Cage

\$2.99   \$2.99   \$5.99   \$2.99   \$3.99

- Scrapped 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     

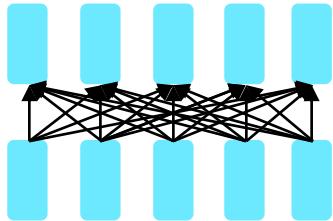


# Lecture Plan

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
  1. Encoders
  2. Encoder-Decoders
  3. Decoders
4. What do we think pretraining is teaching?

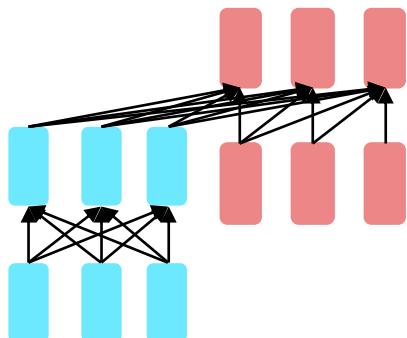
# Pretraining for three types of architectures

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



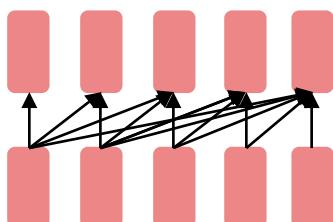
**Encoders**

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

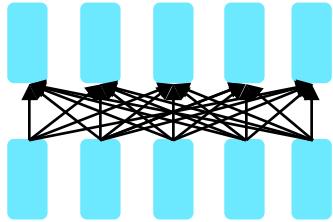


**Decoders**

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

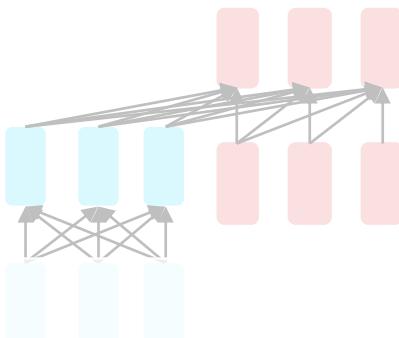
# Pretraining for three types of architectures

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



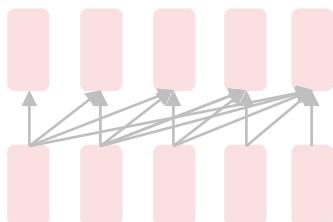
**Encoders**

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



**Encoder-  
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



**Decoders**

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

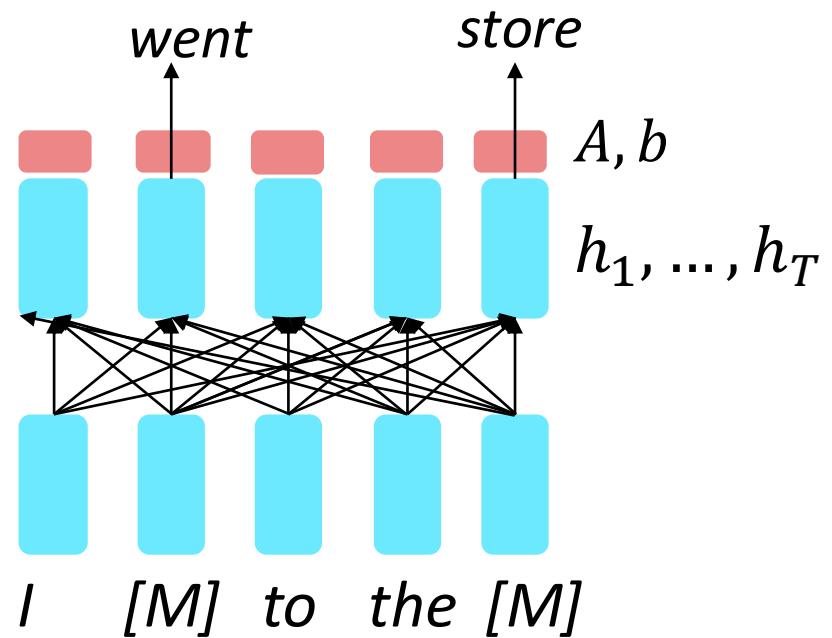
# Pretraining encoders: what pretraining objective to use?

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.

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

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**.



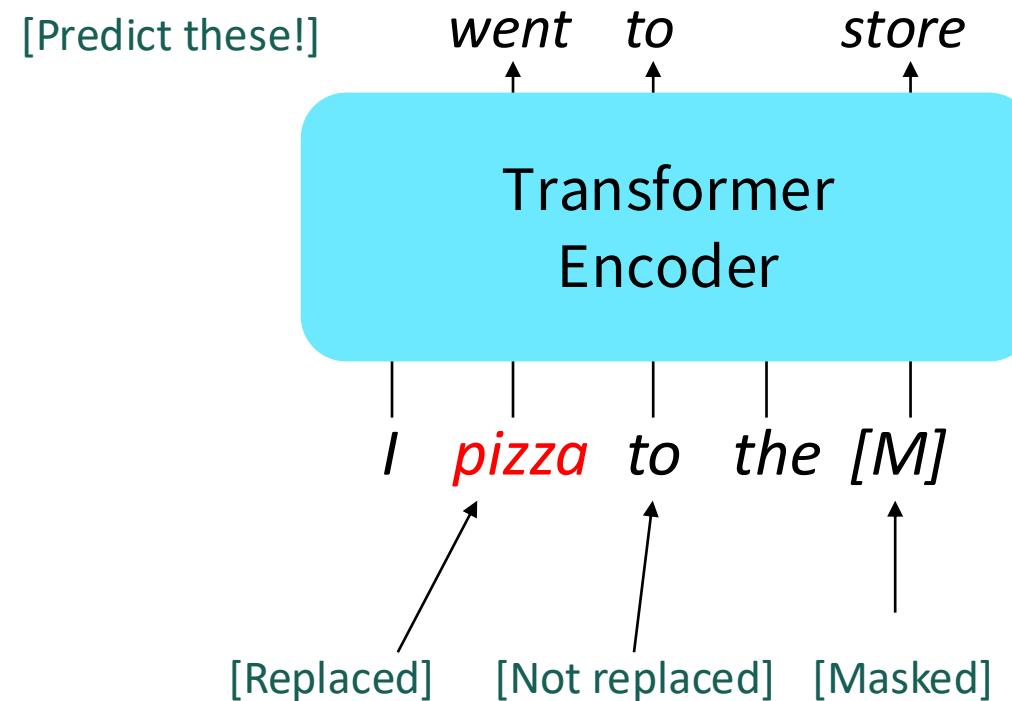
[Devlin et al., 2018]

# BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the “Masked LM” objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

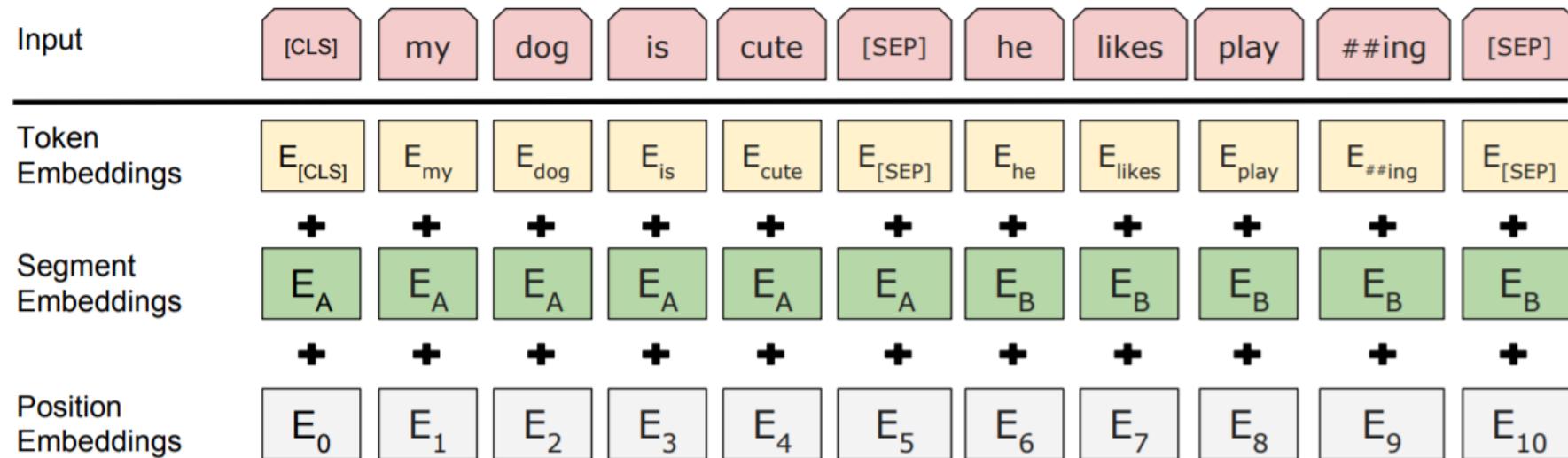
Some more details about Masked LM for BERT:

- 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!)



# 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.

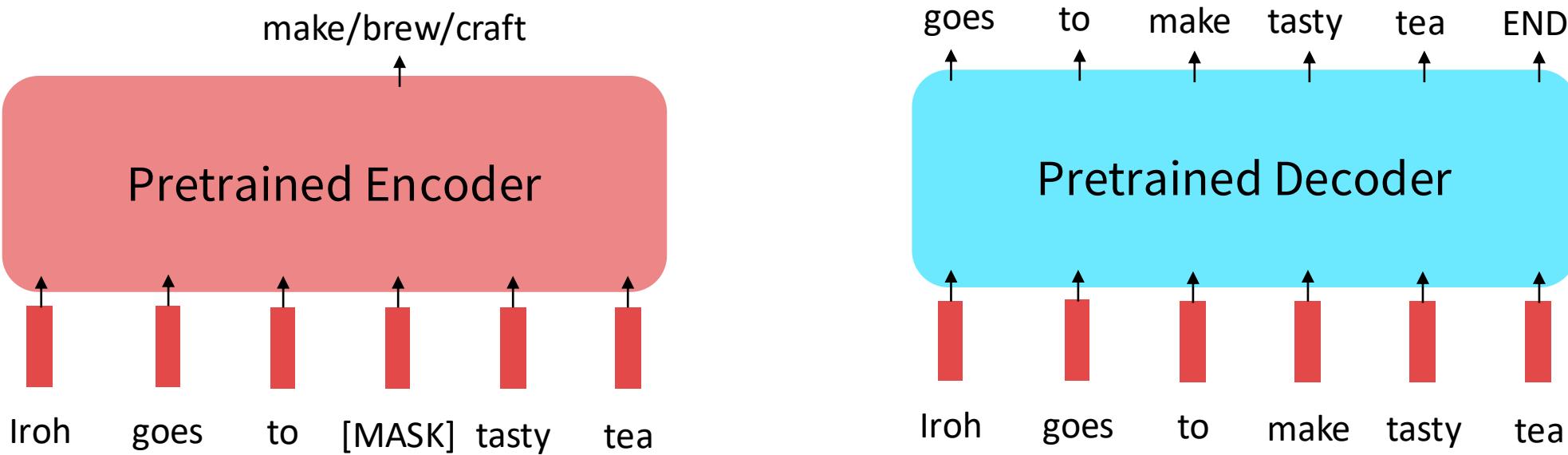
- **QQP**: Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: 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 <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

# 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.

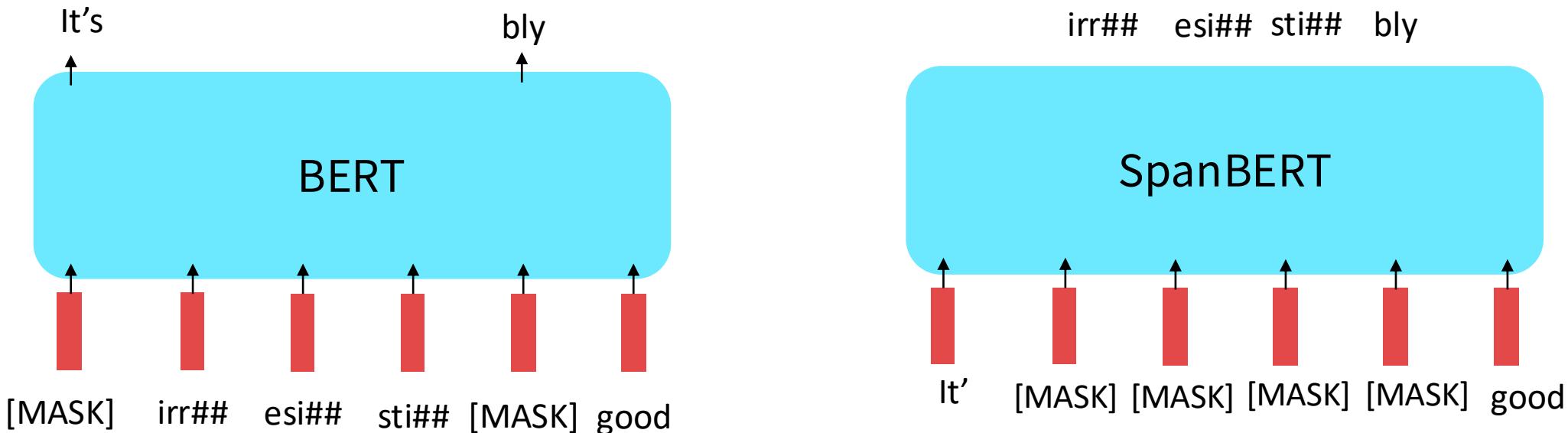


# Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, +++

Some generally accepted improvements to the BERT pretraining formula:

- 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.

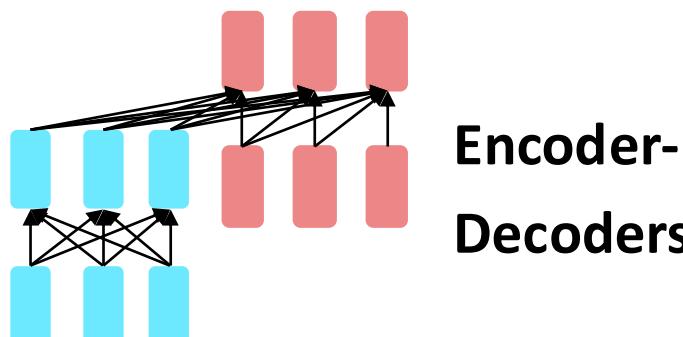
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
<b>RoBERTa</b>						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	<b>94.6/89.4</b>	<b>90.2</b>	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

# Pretraining for three types of architectures

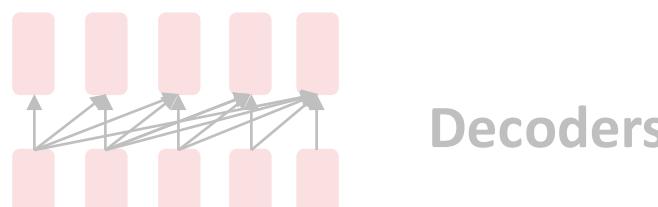
The neural architecture influences the type of pretraining, and natural use cases.



- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



- Good parts of decoders and encoders?
- What's the best way to pretrain them?



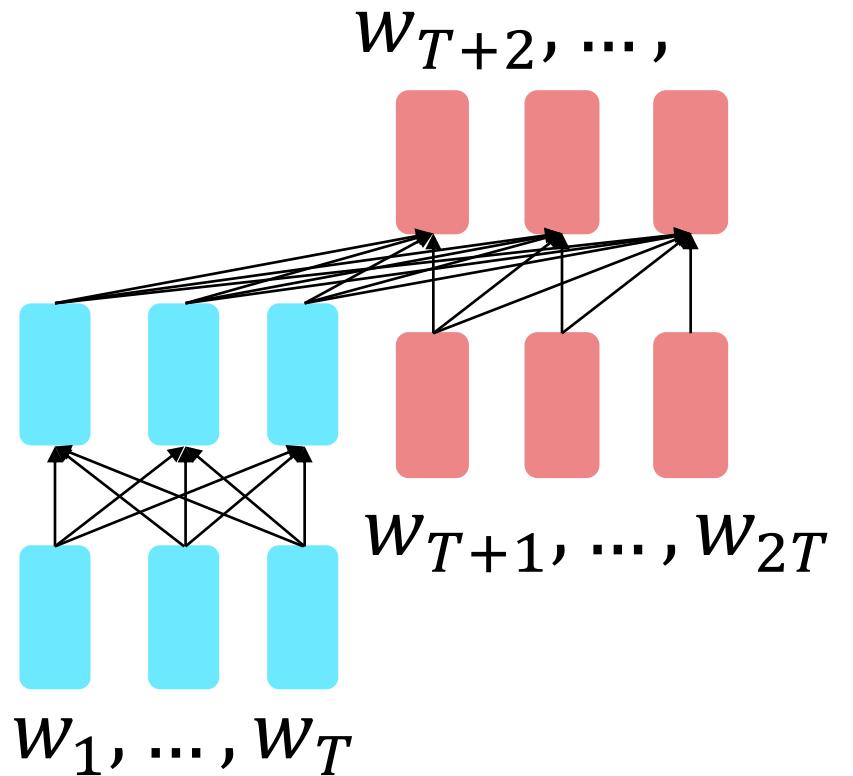
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

# 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.

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

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



[Raffel et al., 2018]

# Pretraining encoder-decoders: what pretraining objective to use?

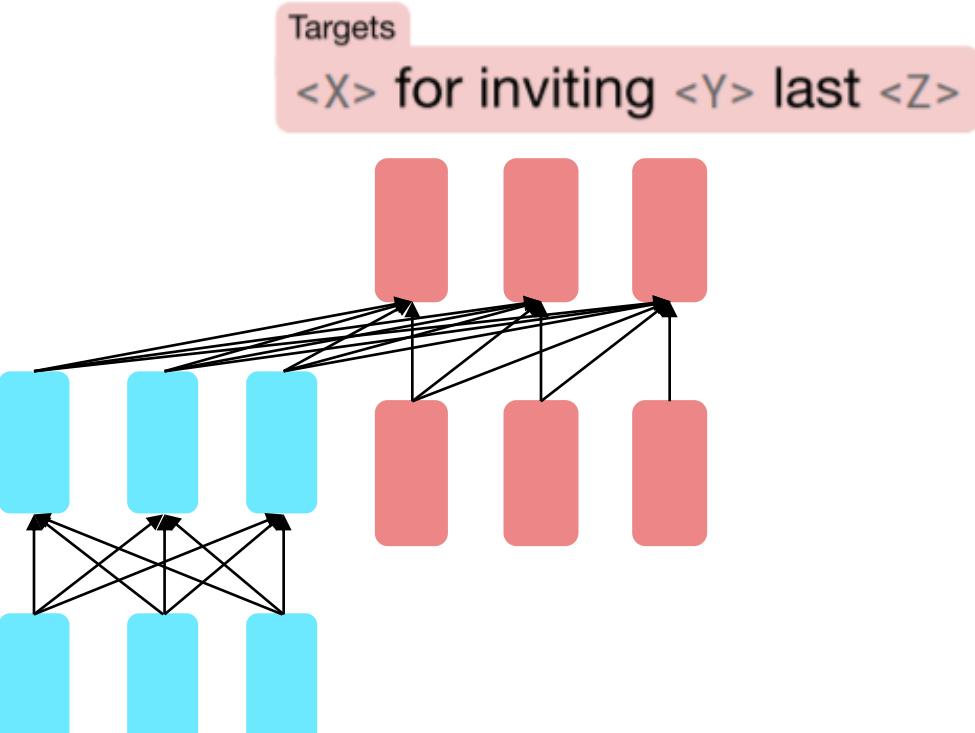
What [Raffel et al., 2018](#) found to work best was **span corruption**. Their model: **T5**.

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.



Inputs

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

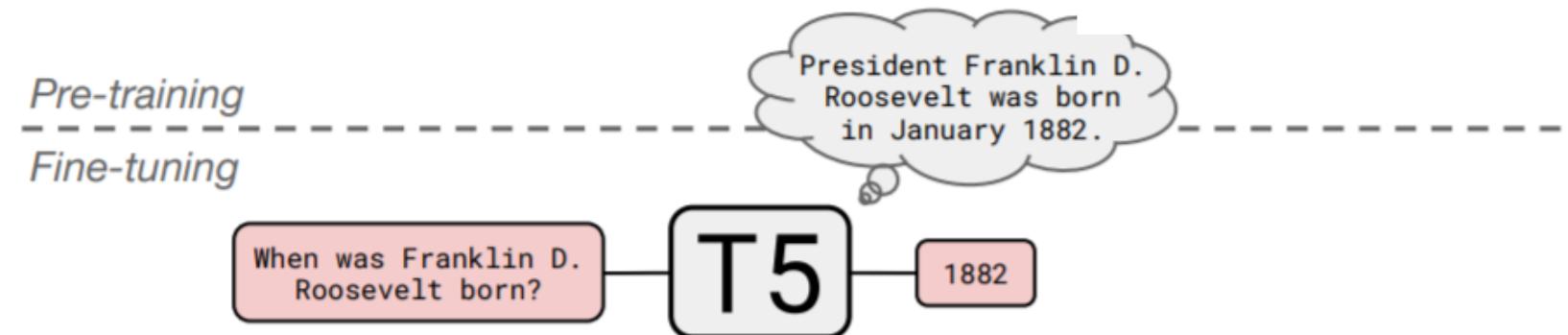
# Pretraining encoder-decoders: what pretraining objective to use?

[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	$2P$	$M$	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
Enc-dec, shared	Denoising	$P$	$M$	82.81	18.78	<b>80.63</b>	<b>70.73</b>	26.72	39.03	<b>27.46</b>
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

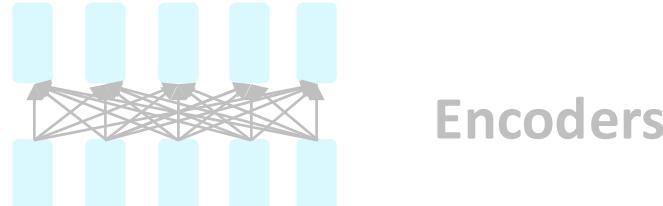
TQA: Trivia QA

All “open-domain” versions

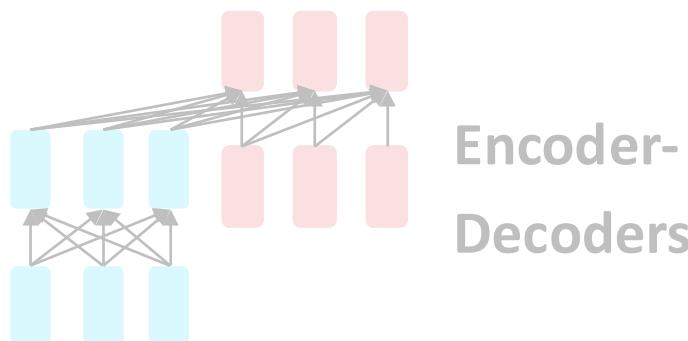
	NQ	WQ	TQA	
			dev	test
Karpukhin et al. (2020)	<b>41.5</b>	42.4	<b>57.9</b>	—
T5.1.1-Base	25.7	28.2	24.2	<b>30.6</b>
T5.1.1-Large	27.3	29.5	28.5	<b>37.2</b>
T5.1.1-XL	29.5	32.4	36.0	<b>45.1</b>
T5.1.1-XXL	32.8	35.6	42.9	<b>52.5</b>
T5.1.1-XXL + SSM	35.2	<b>42.8</b>	51.9	<b>61.6</b>

# Pretraining for three types of architectures

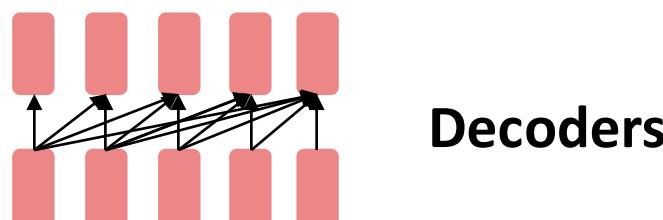
The neural architecture influences the type of pretraining, and natural use cases.



- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



- Good parts of decoders and encoders?
- What's the best way to pretrain them?



- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- All the biggest pretrained models are Decoders.

# Pretraining decoders

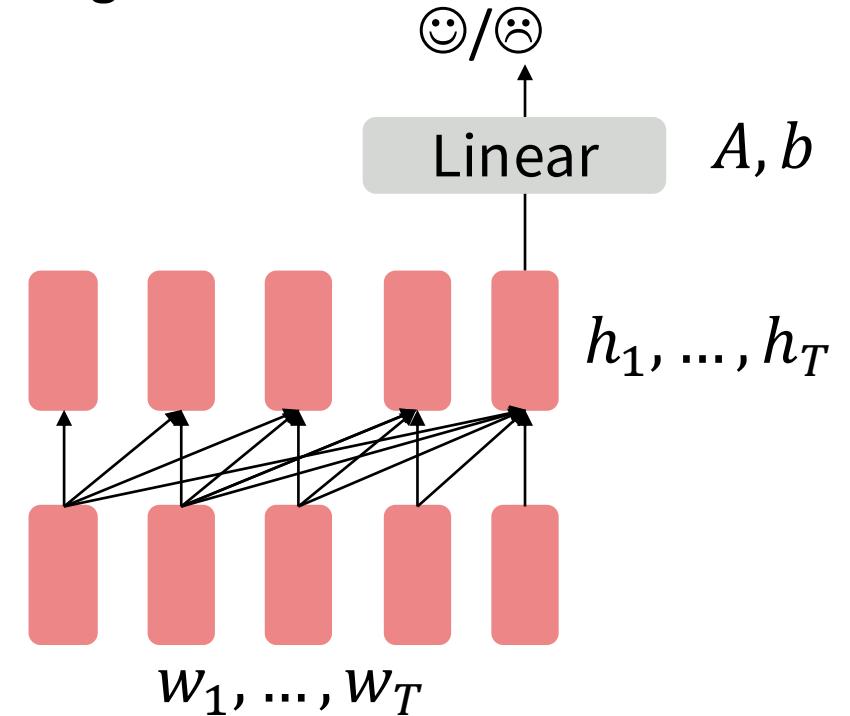
When using language model pretrained decoders, we can ignore that they were trained to model  $p(w_t|w_{1:t-1})$ .

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

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

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

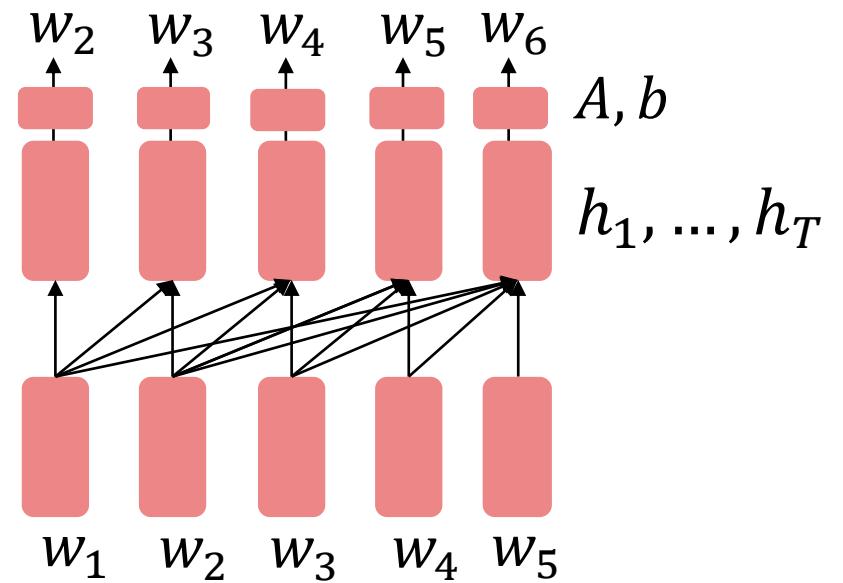
It's natural to pretrain decoders as language models and then use them as generators, finetuning their  $p_\theta(w_t|w_{1:t-1})$ !

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)

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

Where  $A, b$  were pretrained in the language model!



[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, 117M parameters.
- 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.
- The acronym “GPT” never showed up in the original paper; it could stand for “Generative PreTraining” or “Generative Pretrained Transformer”

# 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]

GPT results on various *natural language inference* datasets.

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

# 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.

# 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 take their predictions.

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-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.

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

**Input (prefix within a single Transformer decoder context):**

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

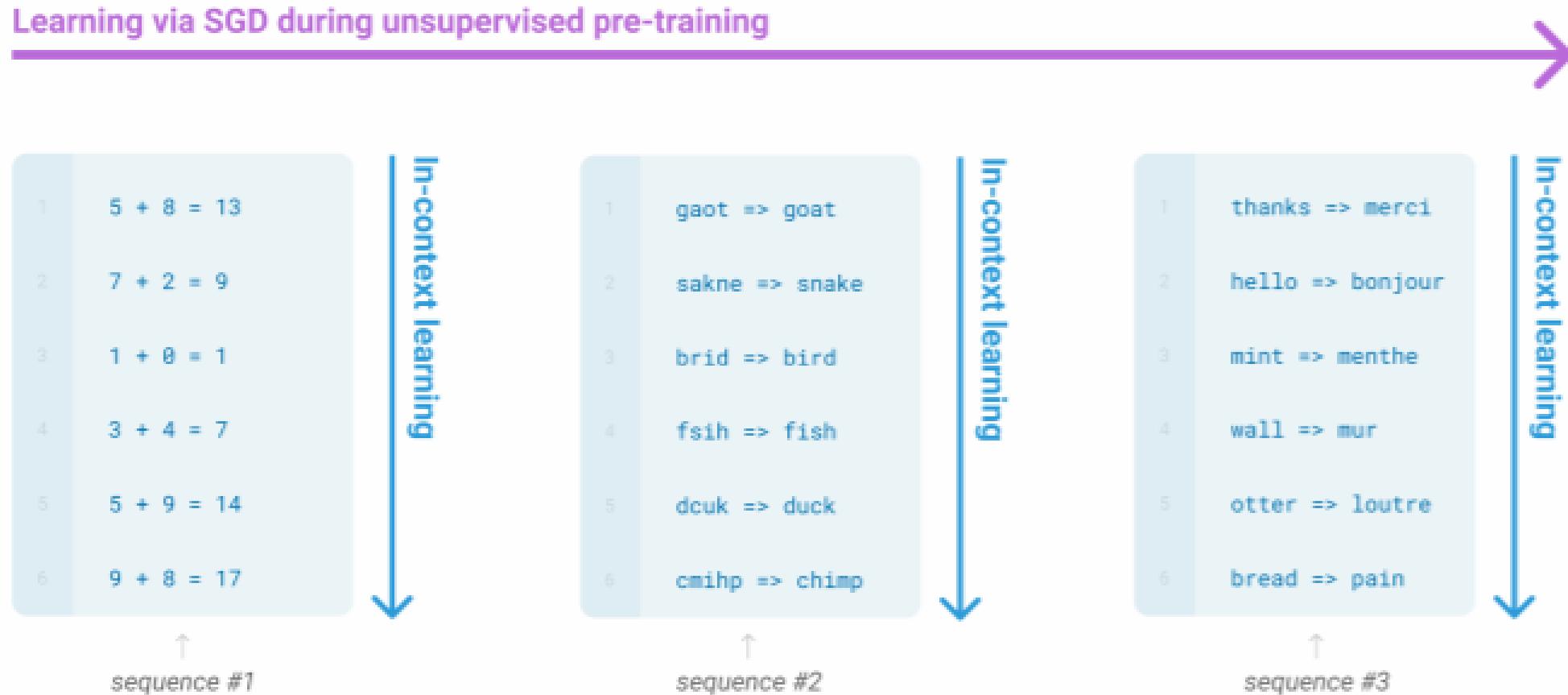
**Output (conditional generations):**

loutre...”

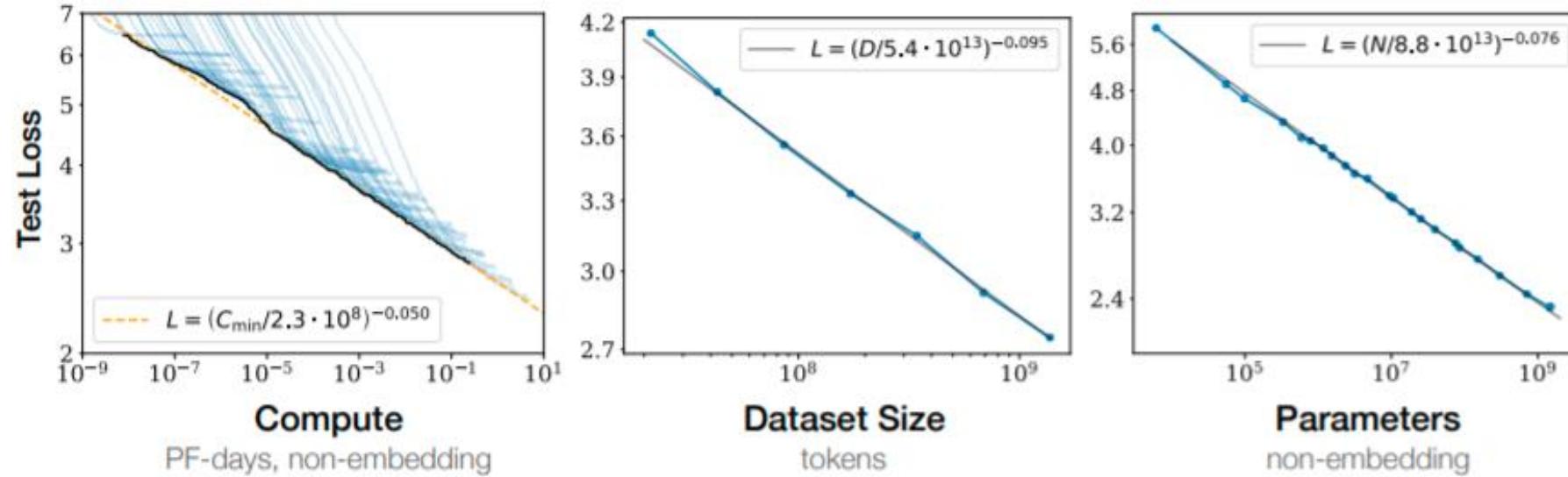
# GPT-3, In-context learning, and very large models

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

Learning via SGD during unsupervised pre-training

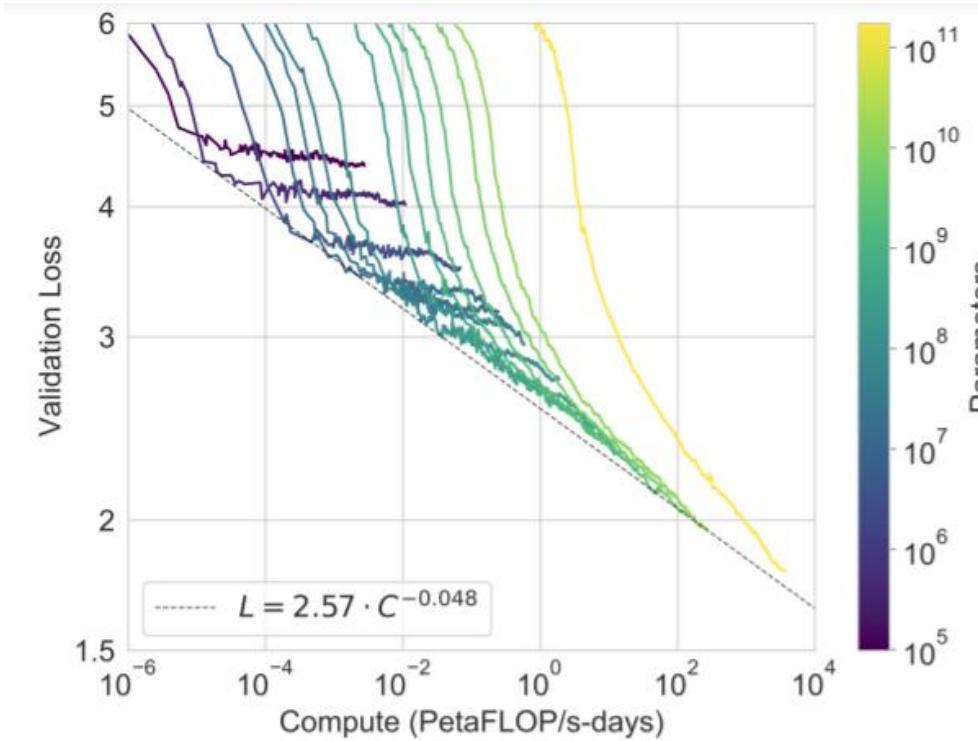


# Why scale? Scaling laws



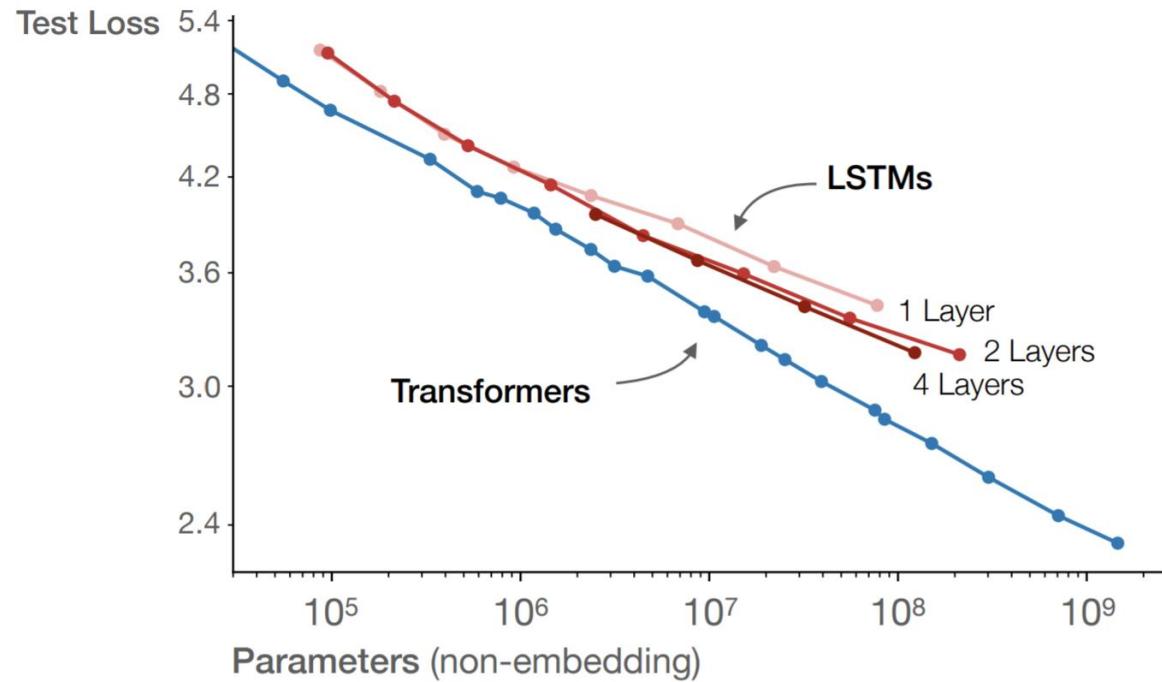
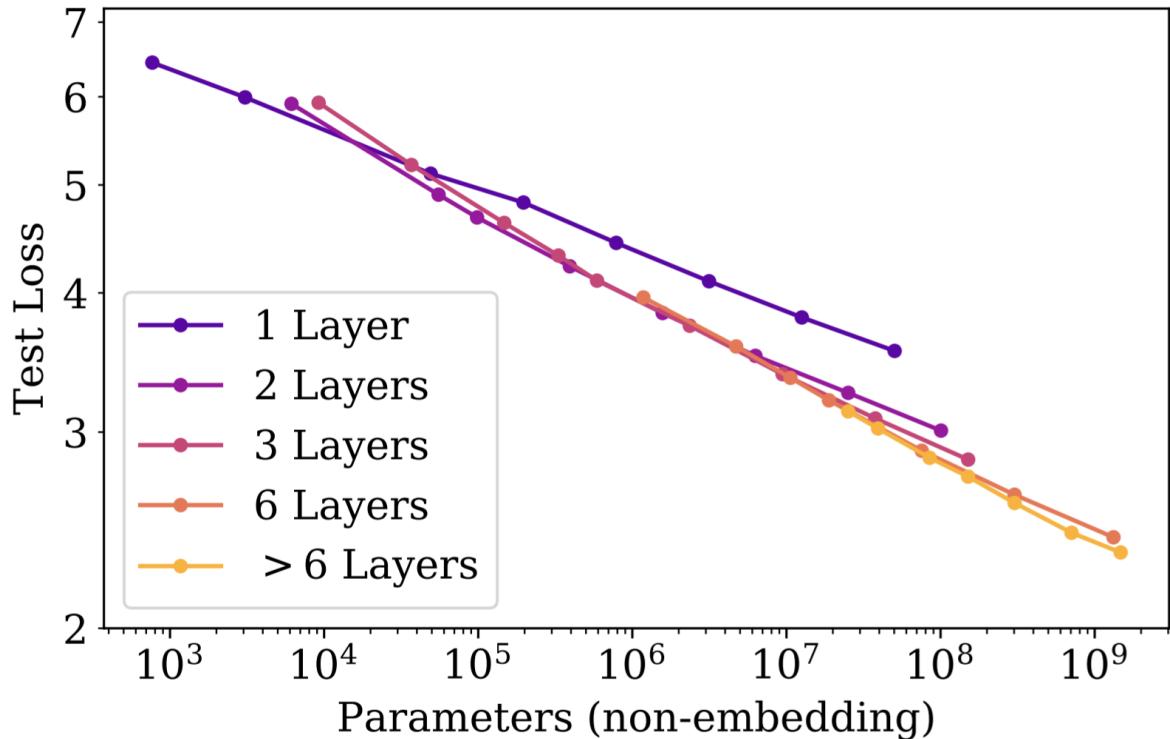
- Empirical observation: scaling up models leads to reliable gains in perplexity

# 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
<i>Chinchilla</i>	70 Billion	1.4 Trillion

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

# Outline

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
  1. Encoders
  2. Encoder-Decoders
  3. Decoders
4. What do we think pretraining is teaching?

# What kinds of things does pretraining teach?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language. Taking our examples from the start of class:

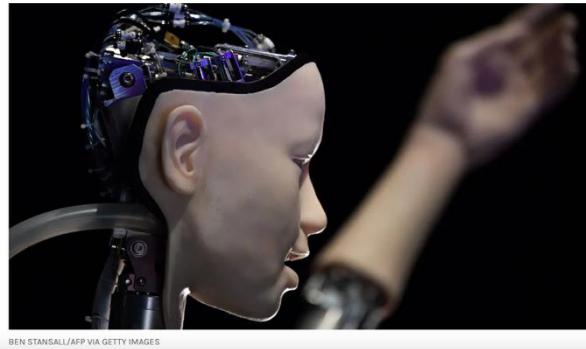
- *Stanford University is located in \_\_\_\_\_, California.* [Trivia]
- *I put \_\_ fork down on the table.* [syntax]
- *The woman walked across the street, checking for traffic over \_\_ shoulder.* [coreference]
- *I went to the ocean to see the fish, turtles, seals, and \_\_\_\_.* [lexical semantics/topic]
- *Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was \_\_.* [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the \_\_\_\_\_. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, \_\_\_\_ [some basic arithmetic; they don't learn the Fibonnaci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

# Sometimes it also memorizes copyrighted material

## AI Art Generators Spark Multiple Copyright Lawsuits

Getty and a trio of artists sued AI art generators in separate suits accusing the companies of copyright infringement for pilfering their works.

BY WINSTON CHO  JANUARY 17, 2023 4:10PM

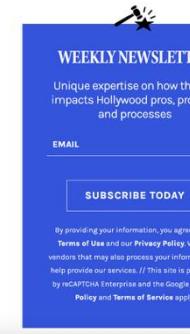


BEN STANSALL/AFP VIA GETTY IMAGES

## Anthropic fires back at music publishers' AI copyright lawsuit

By Blake Brittain

January 17, 2024 3:30 PM PST · Updated 19 days ago



## ARTICLE Insights from the Pending Copilot Class Action Lawsuit

October 4, 2023

Bloomberg Law

By Daniel R. Mello, Jr.; Jenevieve J. Maerker; Matthew C. Berntsen; Ming-Tao Yang

GitHub Inc. offers a cloud-based platform that is popular among many software programmers for hosting and sharing source code, and collaborating on source code drafting. GitHub's artificial

## *The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work*

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.



# Sometimes it learns some things we don't want..

- *Membership inference* lets you recover parts of the training data
- Sometimes this training data is semi-private material from the web (addresses, emails)

