PURDUE UNIVERSITY NORTHWEST

Natural Language Processing
CS 52570

Lecture 17 ----- Pre-training

Prof. Yang Ni @ CS Department, PNW

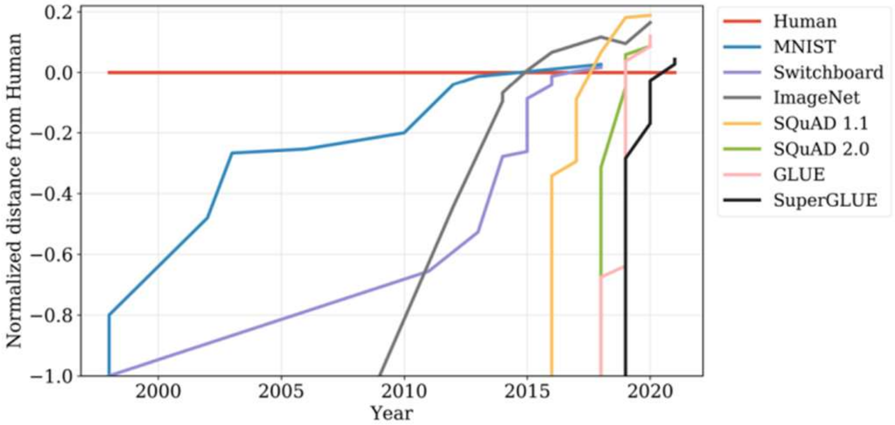
Part of this course is derived using resources from Stanford CS 224N, originated by Dr. Manning and others.

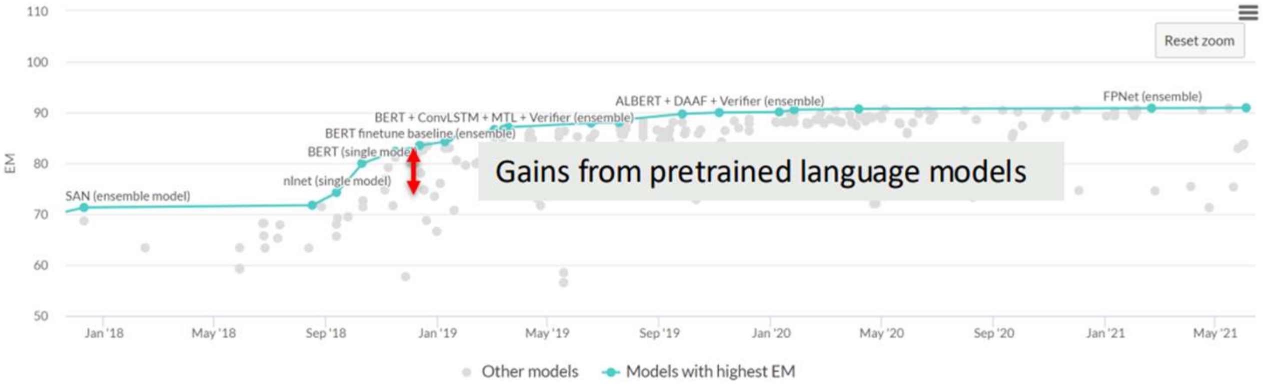
Lecture Plan

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
   - Decoders
   - Encoders
   - Encoder-Decoders
4. Interlude: what do we think pretraining is teaching?
5. Very large models and in-context learning

The pretraining revolution

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

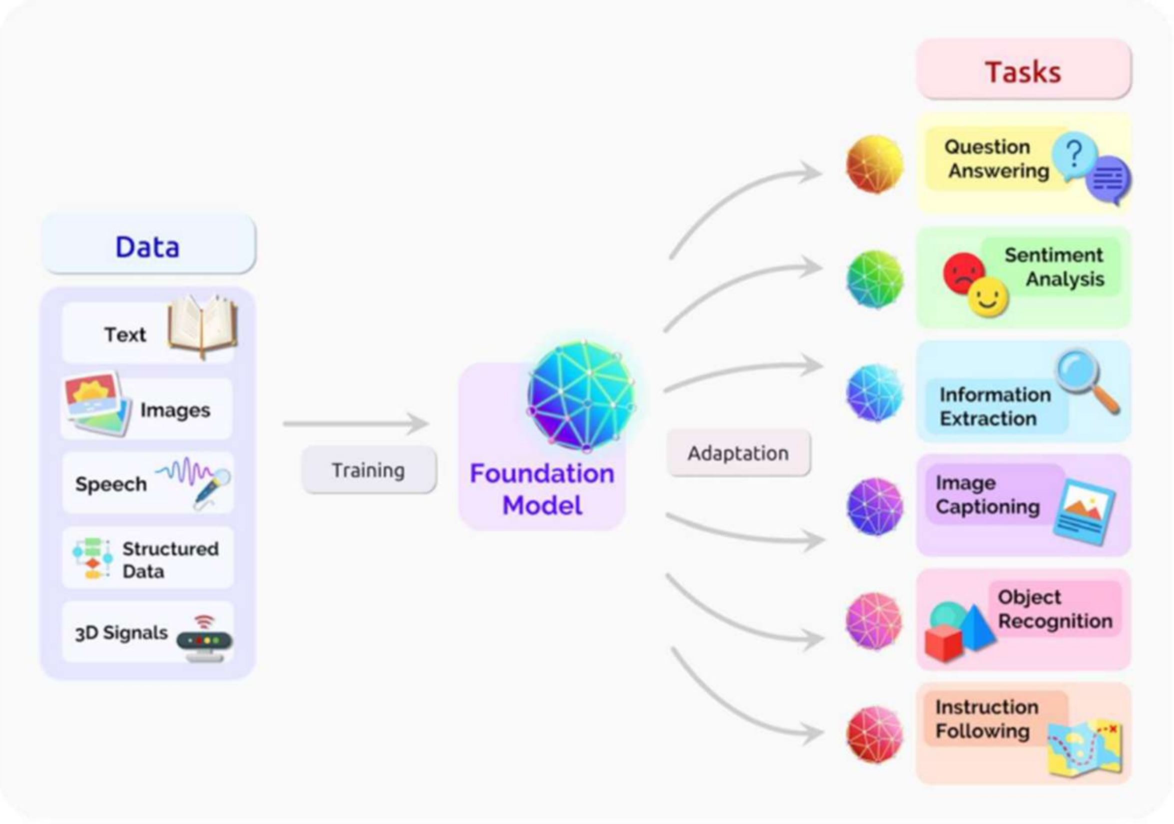




Gains from pretrained language models

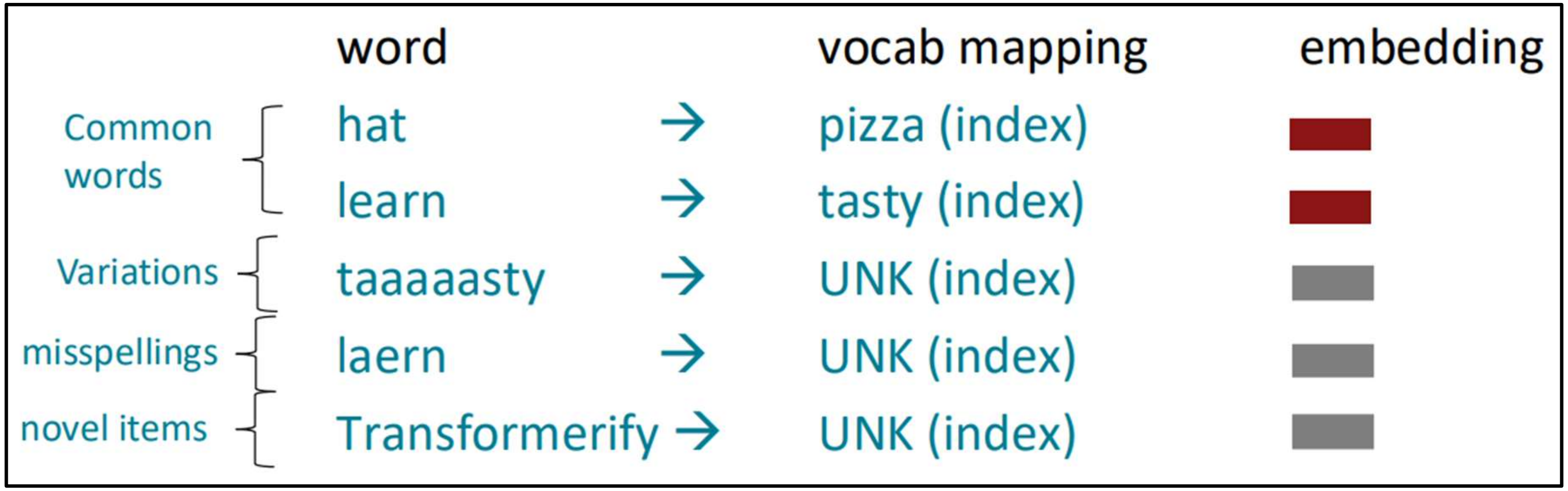
Pretraining
– scaling unsupervised learning on the internet

- 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)	▬
misspellings	laern	→	UNK (index)	▬
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.

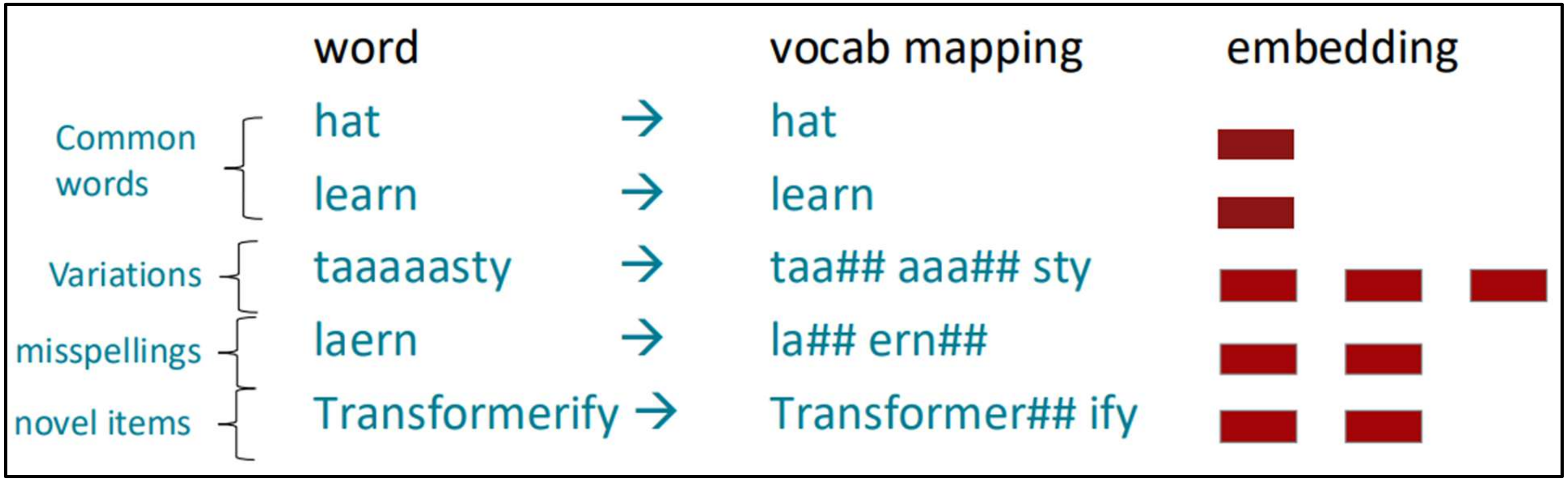
Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary.

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

Lecture Plan

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
   - Decoders
   - Encoders
   - Encoder-Decoders
4. Interlude: what do we think pretraining is teaching?
5. 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)

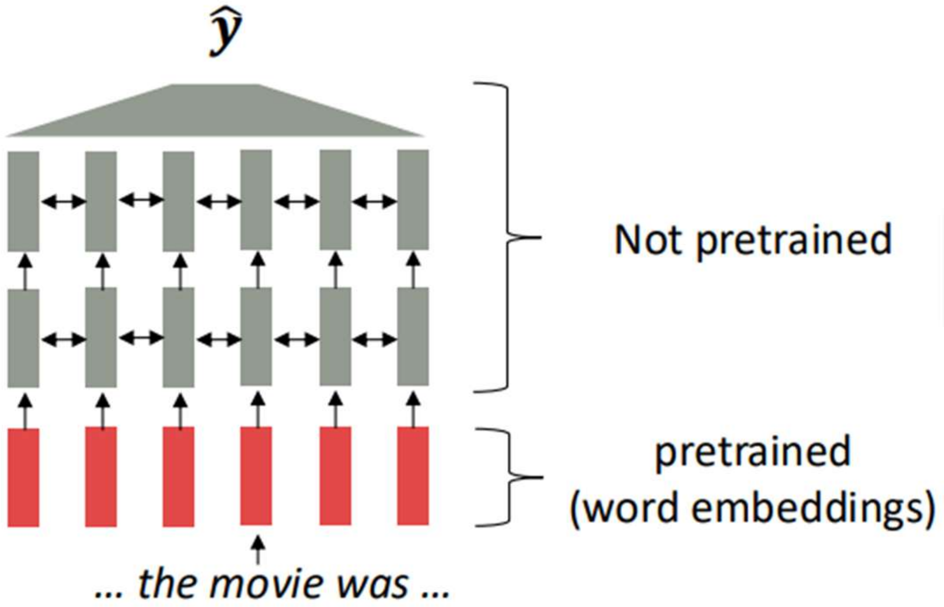
- This quote is a summary of distributional semantics, and motivated word2vec. But:

  "… 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.

Where we were: pretrained word embeddings

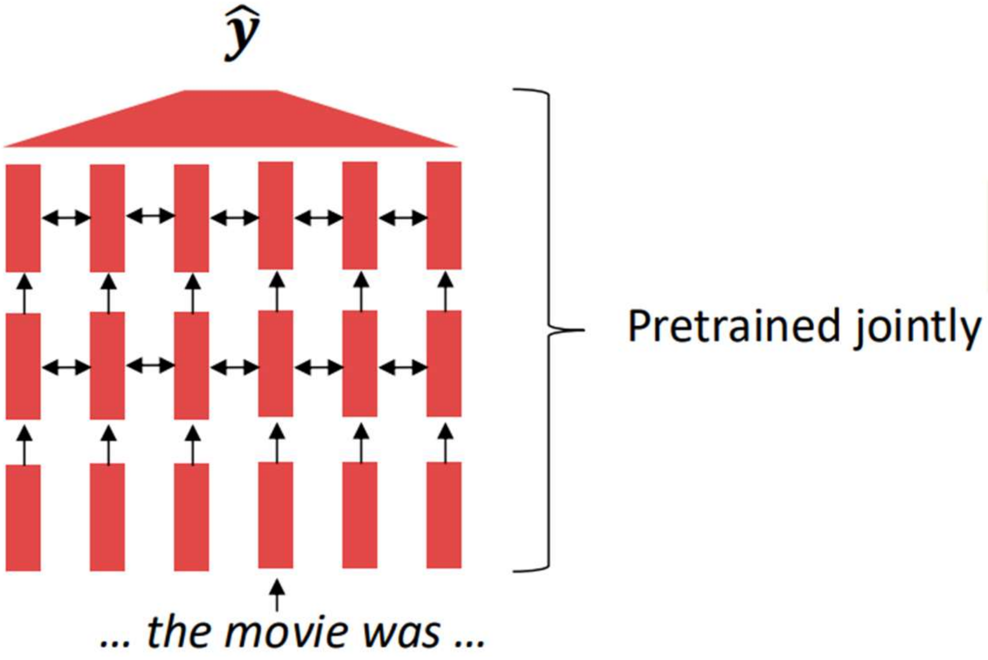
- ➢ Circa 2017:
- Start with pretrained word embeddings (no context!)
- Learn how to incorporate context in an LSTM or Transformer while training on the task.
- ➢ Some issues to think about:
- The training data we have for our downstream task (like question answering) must be sufficient to teach all 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?

Purdue University Northwest is located in ___________, Indiana.

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?

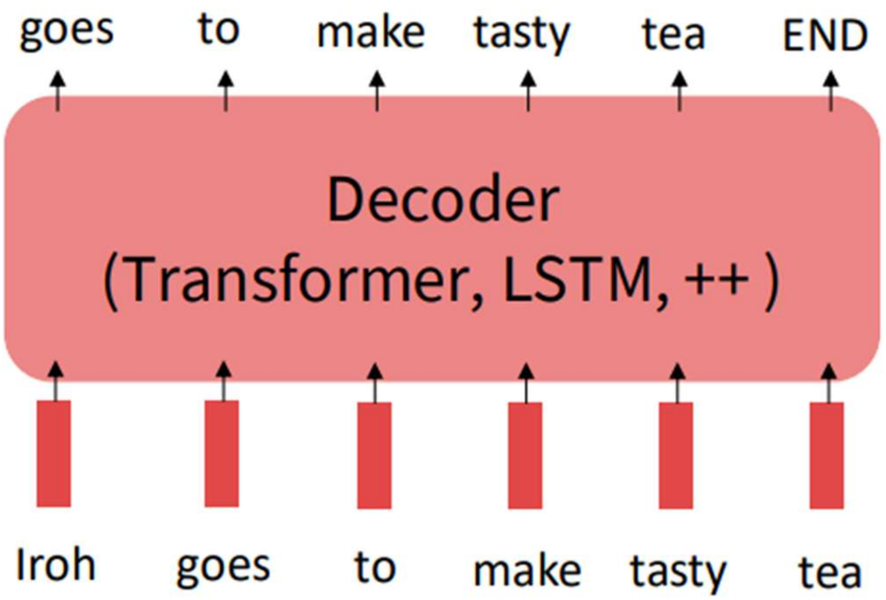
Iroh went into the kitchen to make some tea.
Standing next to Iroh, Zuko pondered his destiny.
Zuko left the ______.

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]

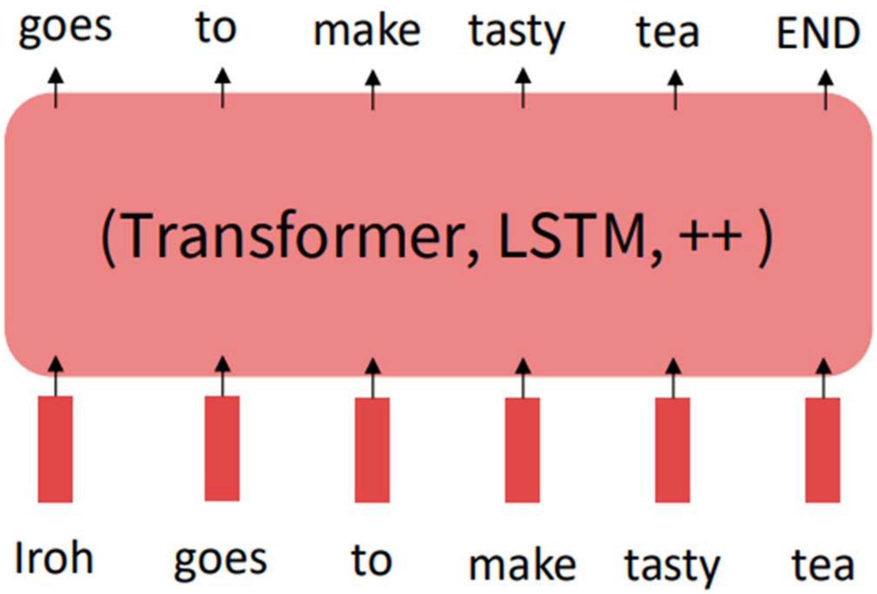
- Recall the **language modeling** task:
- Model $P_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- There’s lots of data for this! (In English.)
- Pretraining through language modeling:
- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



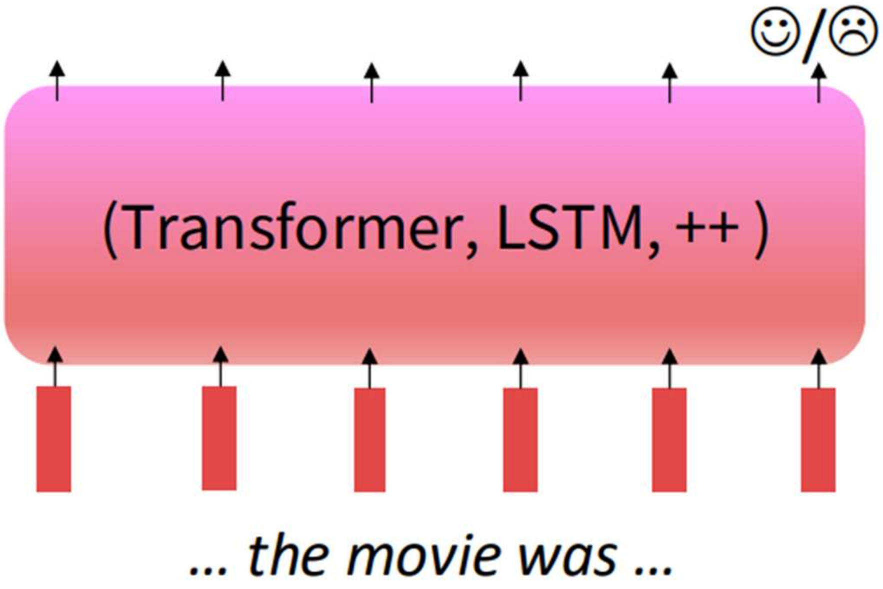
The Pretraining / Finetuning Paradigm

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!



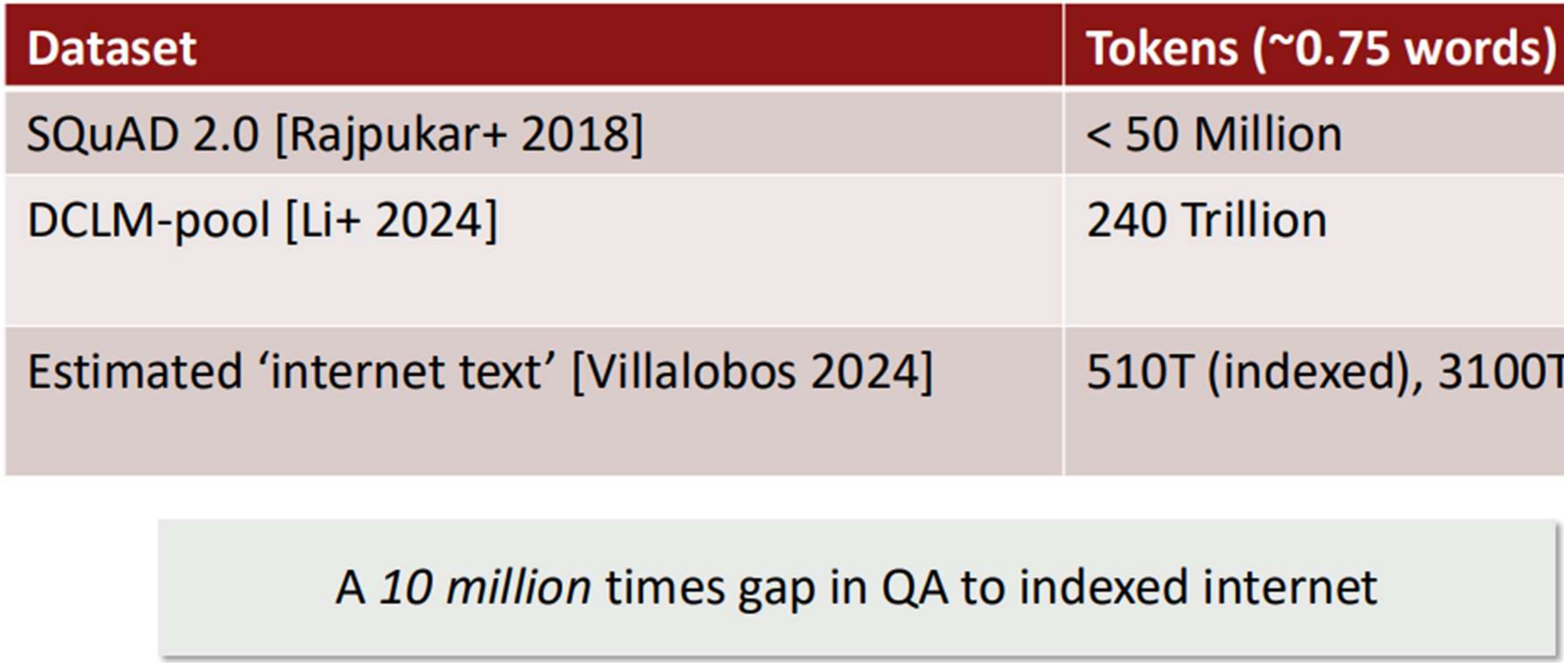
Stochastic gradient descent and pretrain/finetune

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

- Consider, provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta)$.
  (The pretraining loss.)
- Then, finetuning approximates $\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta)$., , 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



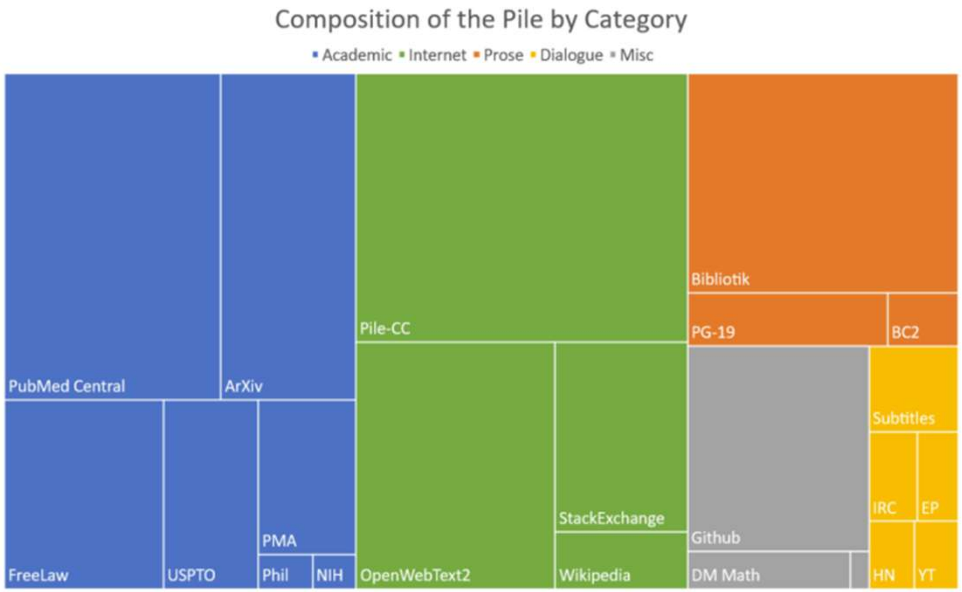
Dataset	Tokens (~0.75 words)
SQuAD 2.0 [Rajpukar+ 2018]	< 50 Million
DCLM-pool [Li+ 2024]	240 Trillion
Estimated 'internet text' [Villalobos 2024]	510T (indexed), 3100T (total)

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

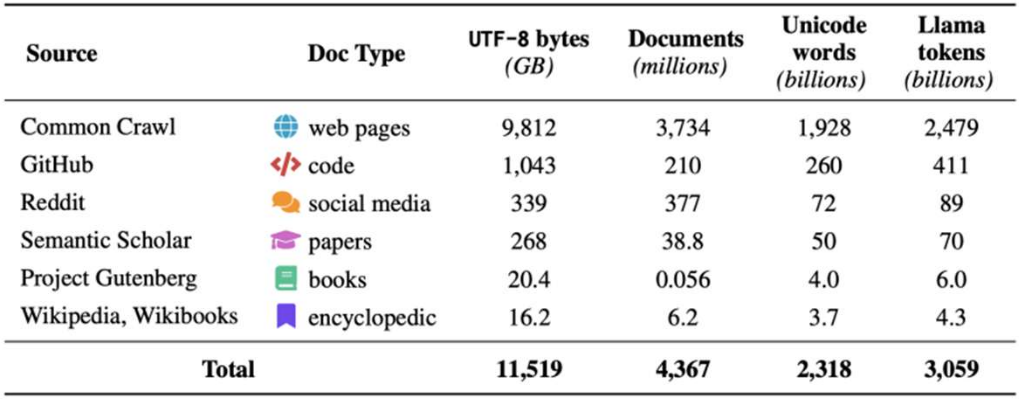
With this much data, we might make progress on even the hardest fill-in-the-blank tasks.

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)

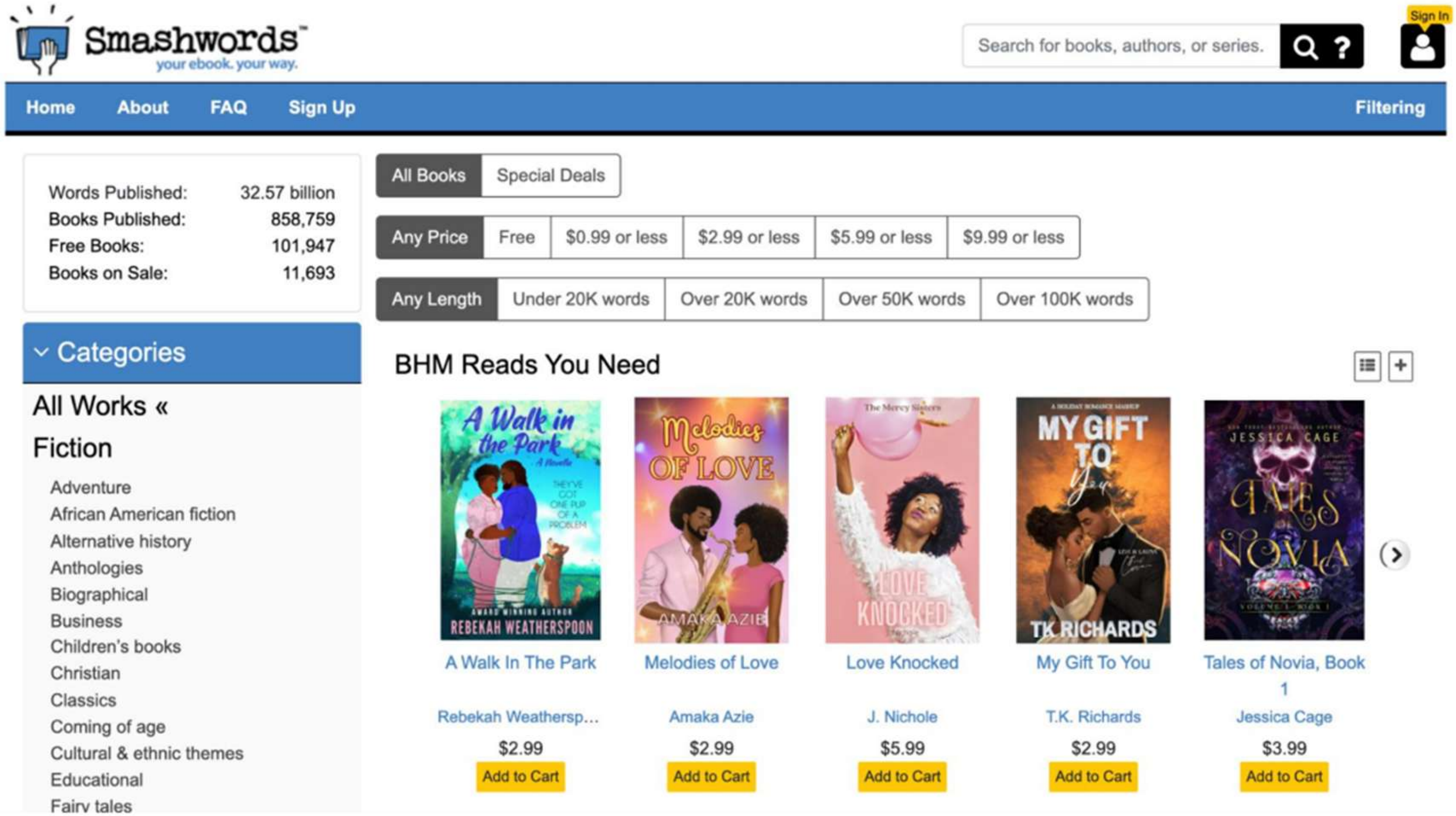
- *Book Title Poetry: April 26,\u00a02012\n\n\nCan't recall where I first saw this but I'll admit it isn't my own original idea. I think it was a link on Twitter or something, which I found whilst poking around. Of course I can't find it again.\n\nBasically, the idea is you grab a few books from a shelf 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/

- *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..



Smashwords
your ebook. your way.

Search for books, authors, or series.

Sign In

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

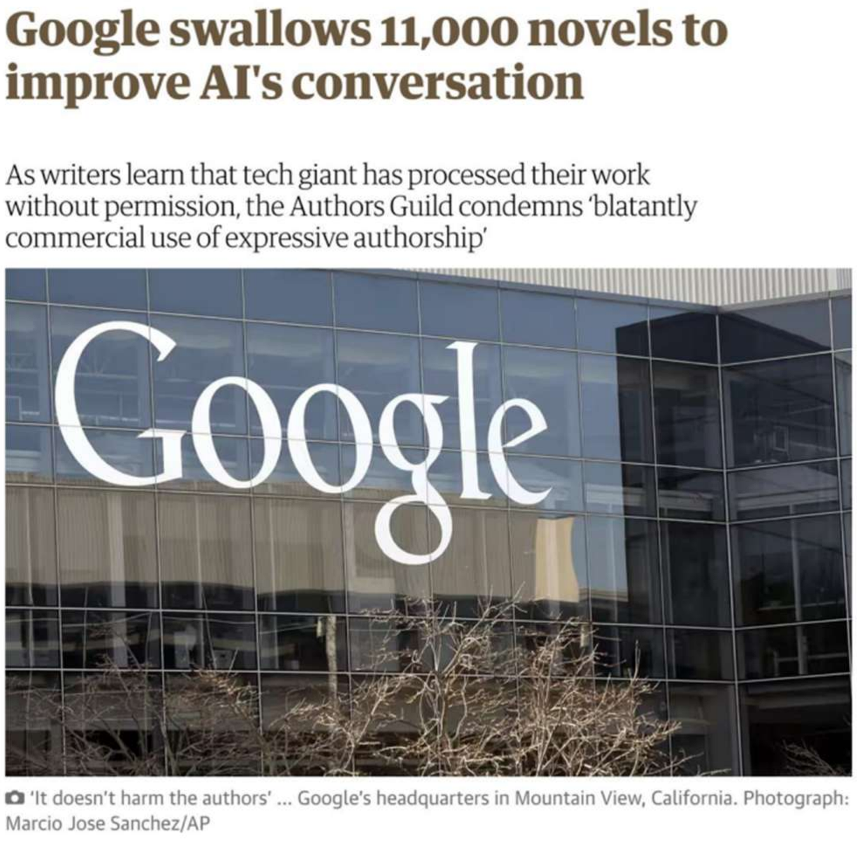
- Adventure
- African American fiction
- Alternative history
- Anthologies
- Biographical
- Business
- Children's books
- Christian
- Classics
- Coming of age
- Cultural & ethnic themes
- Educational
- Fairy tales

BHM Reads You Need

Title	Author	Price	
A Walk In The Park	Rebekah Weathersp...	$2.99	Add to Cart
Melodies of Love	Amaka Azie	$2.99	Add to Cart
Love Knocked	J. Nichole	$5.99	Add to Cart
My Gift To You	T.K. Richards	$2.99	Add to Cart
Tales of Novia, Book 1	Jessica Cage	$3.99	Add to Cart

Scraped ebooks from the internet – highly controversial

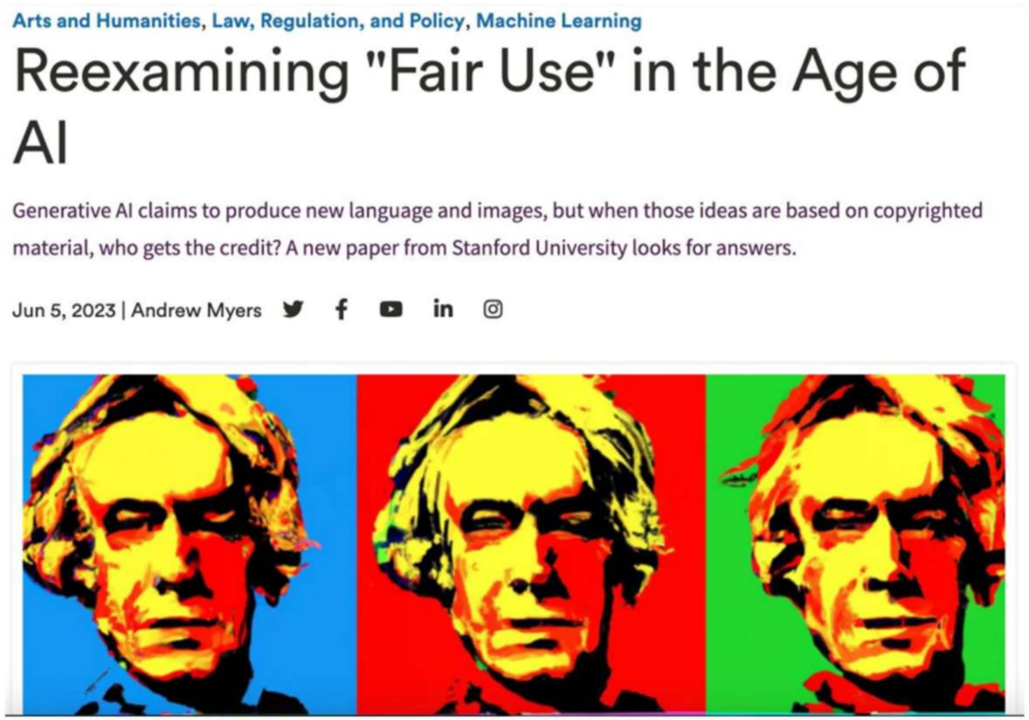
Fair use and other concerns



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.

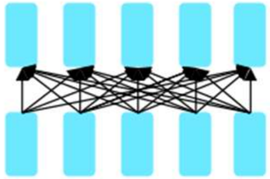
Jun 5, 2023 | Andrew Myers

Lecture Plan

1. A brief note on subword modeling
2. Motivating model pretraining from word embeddings
3. Model pretraining three ways
   - Encoders
   - Encoder-Decoders
   - Decoders
4. Interlude: what do we think pretraining is teaching?
5. Very large models and in-context learning

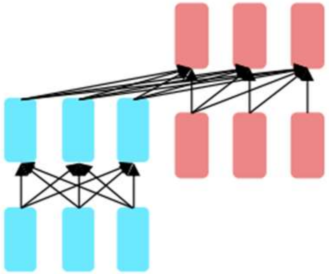
Pretraining for three types of architectures

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



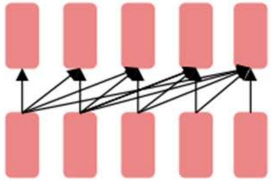
Encoders

- 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?

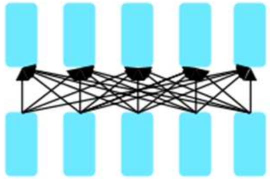


Decoders

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

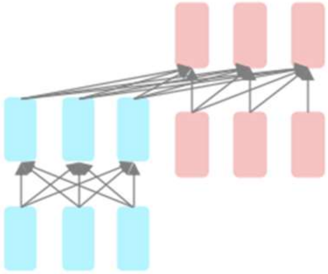
Pretraining for three types of architectures

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



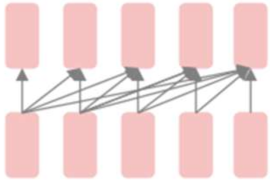
Encoders

- 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.

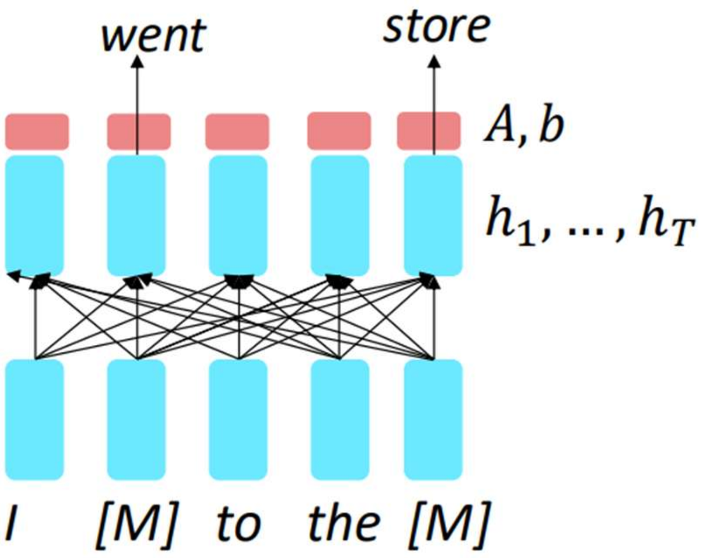
Pretraining encoders: what pretraining objective to use?

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**.



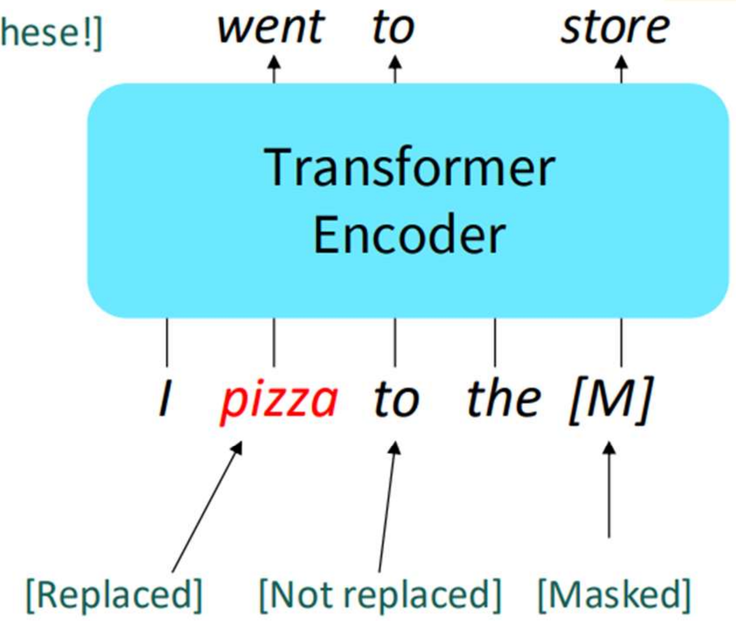
[Devlin et al., 2018]

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:

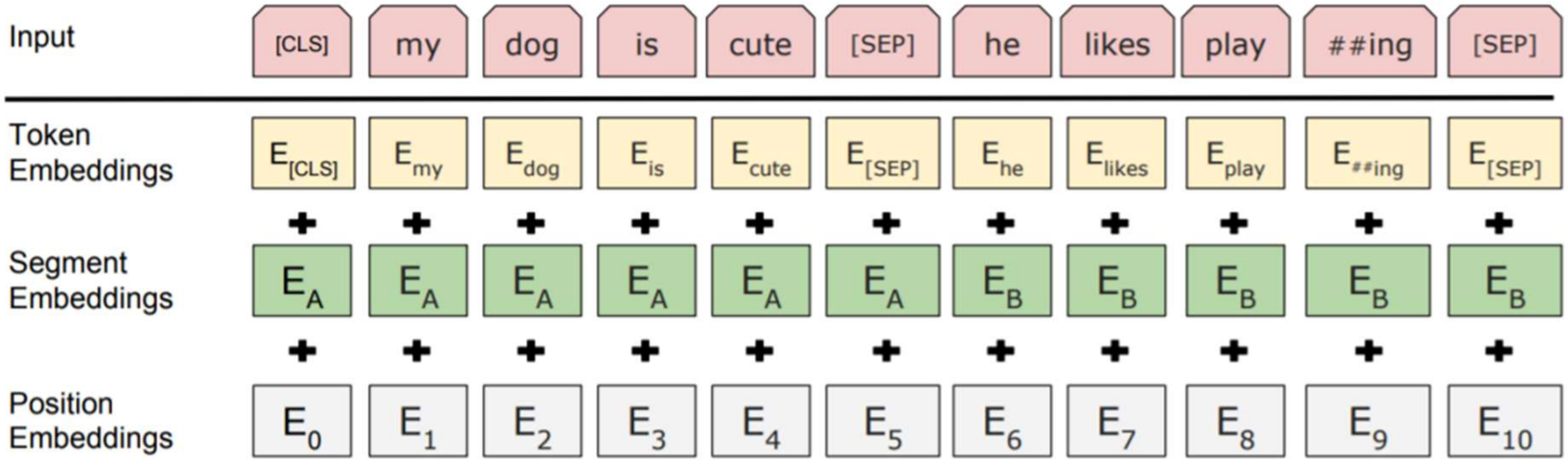
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time.
  - Replace input word with a random token 10% of the time.
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations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



[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

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

[Devlin et al., 2018, Liu et al., 2019]

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.”

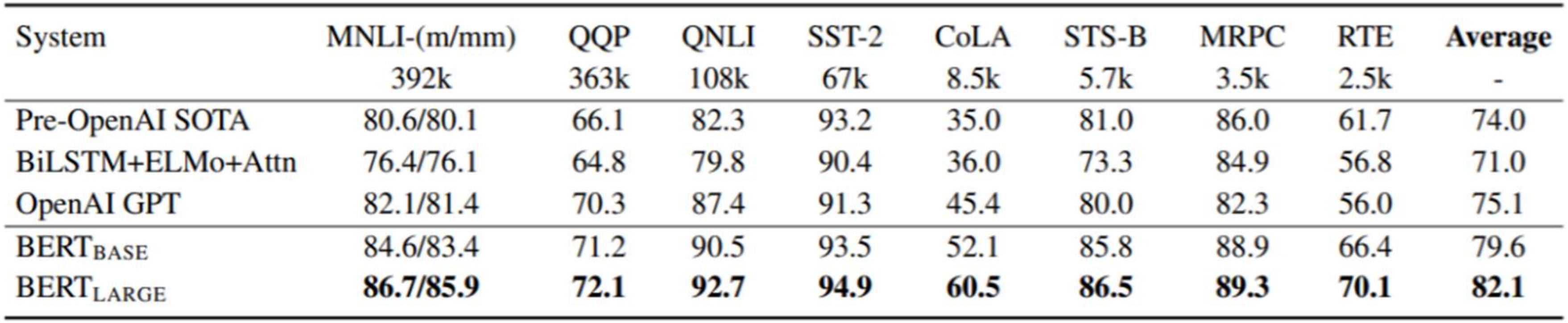
[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

BERT was massively popular and hugely versatile; finetuning BERT led to new state-ofthe-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

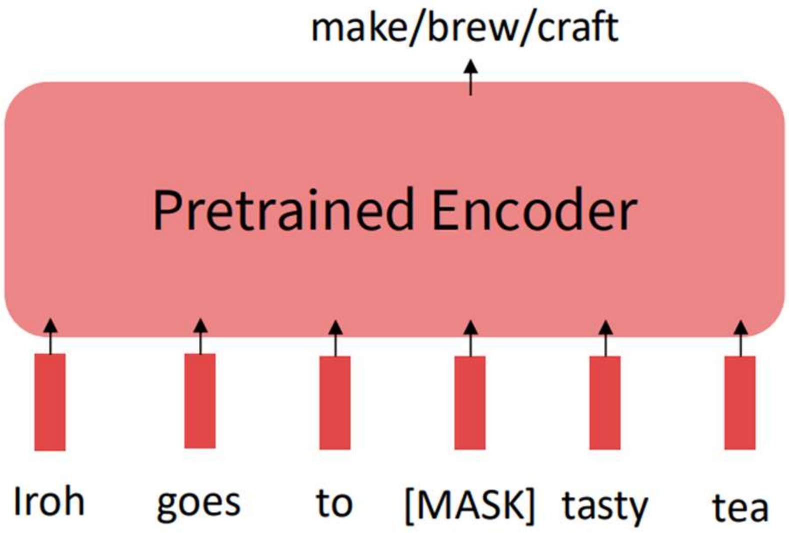


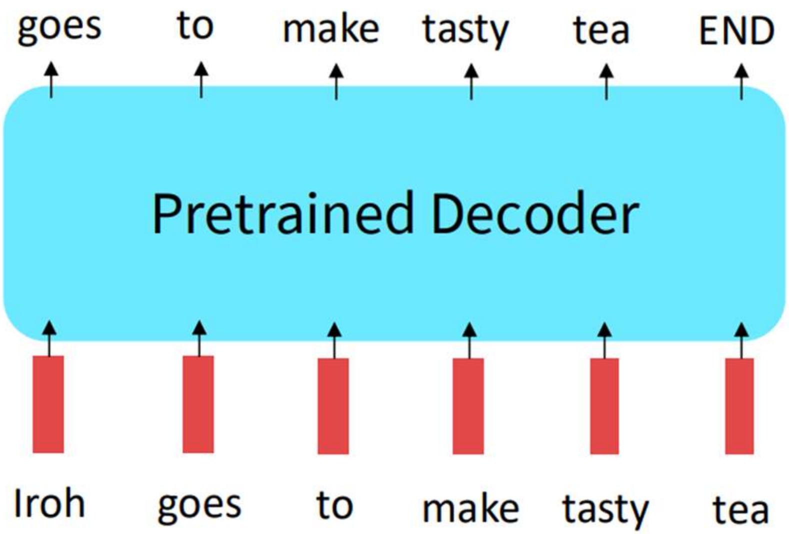
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1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

[Devlin et al., 2018]

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.



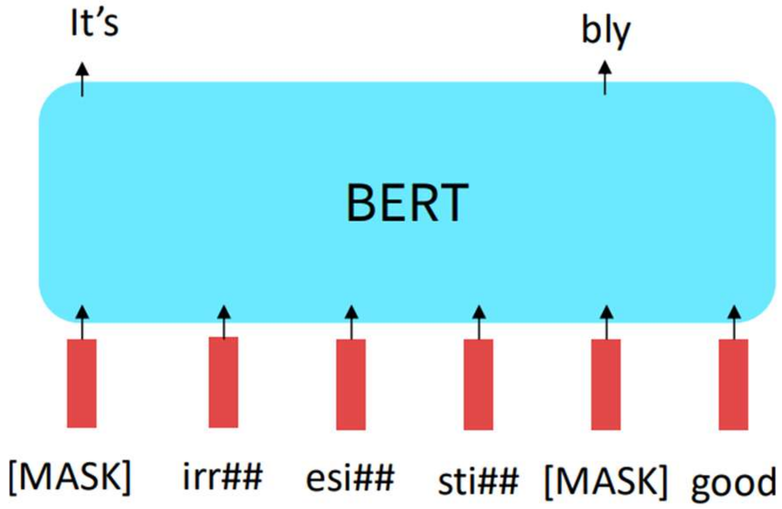


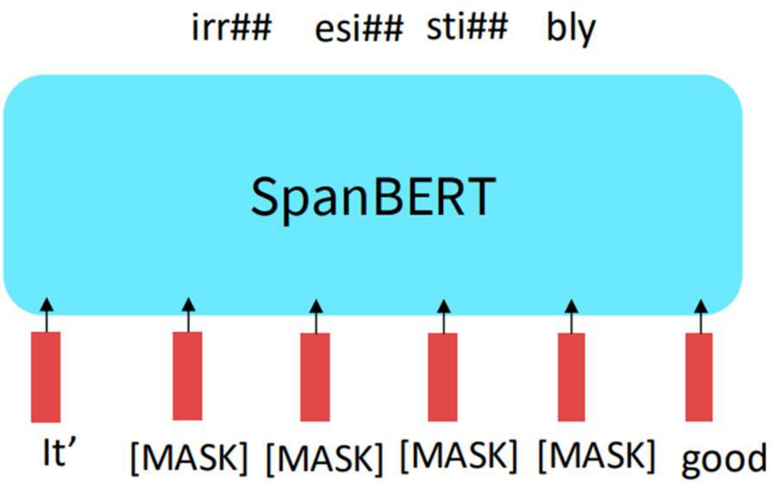
Extensions of BERT

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

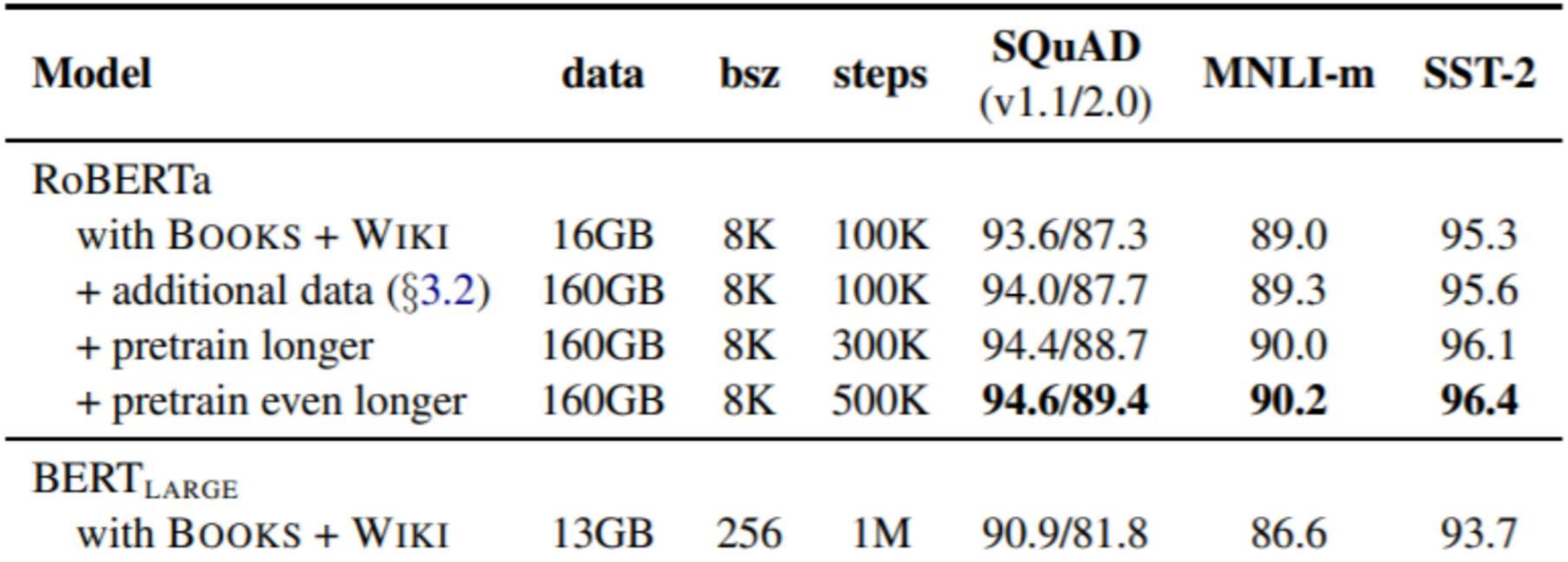




[Liu et al., 2019; Joshi et al., 2020]

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**
BERT$_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

[Liu et al., 2019; Joshi et al., 2020]

Full Finetuning vs. Parameter-Efficient Finetuning

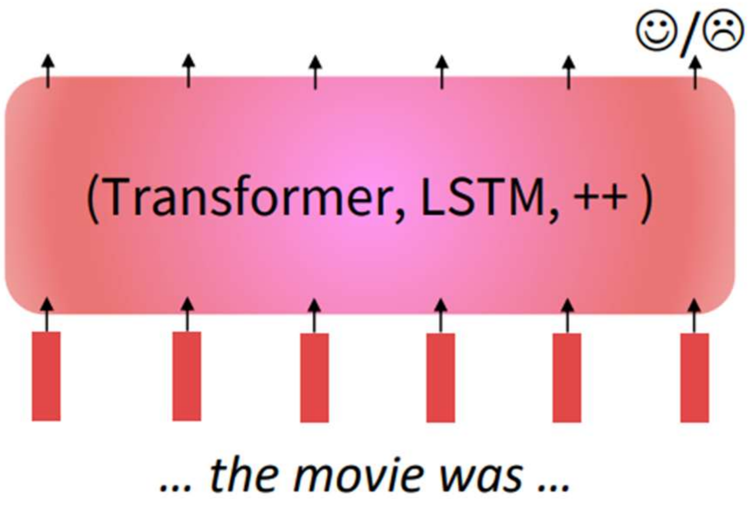
Finetuning every parameter in a pretrained model is **memory-intensive**.

But **lightweight finetuning** methods adapt pretrained models in a constrained way.

Leads to **less overfitting** and/or **more efficient finetuning and inference**.

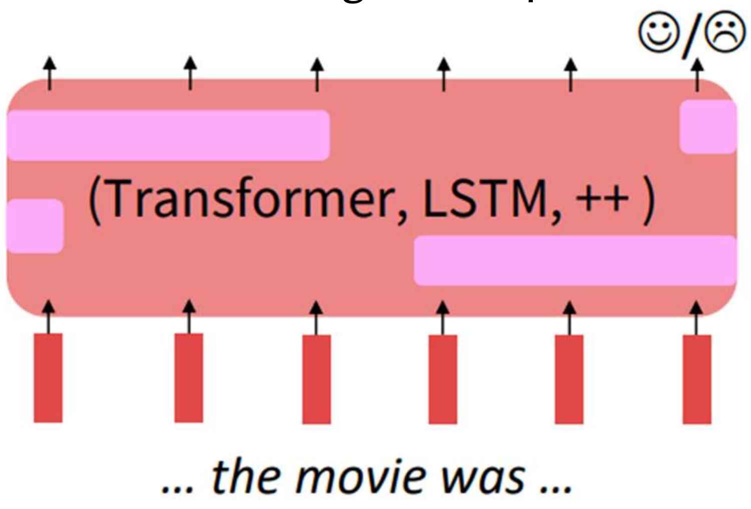
Full Finetuning

Adapt all parameters



Lightweight Finetuning

Train a few existing or new parameters



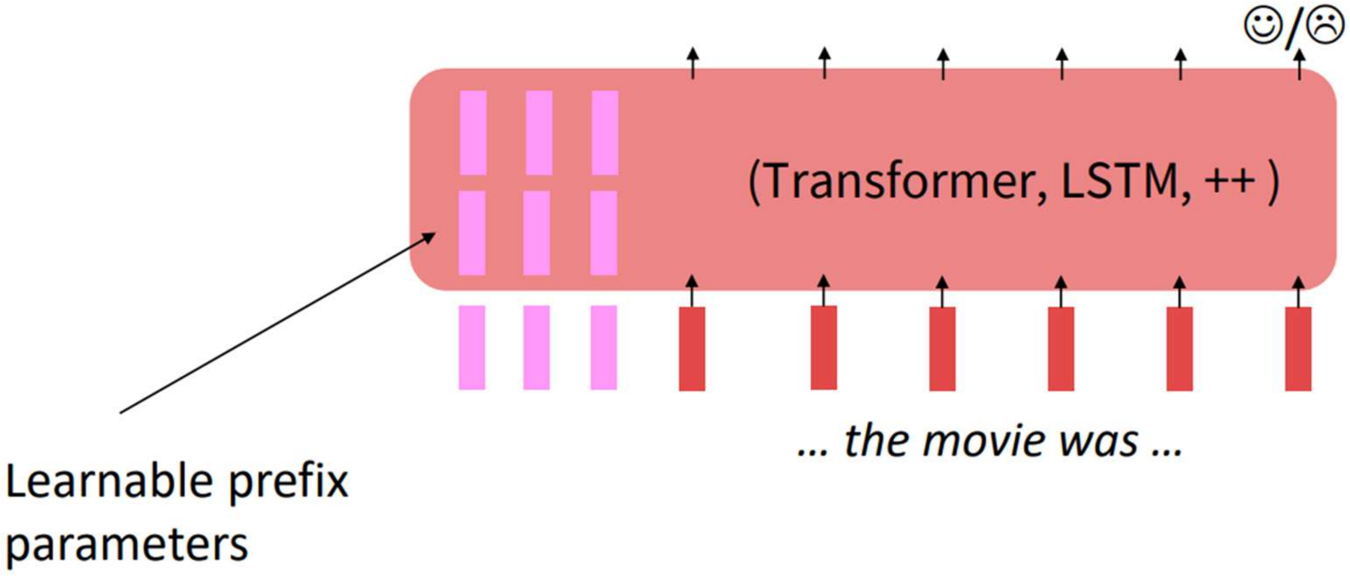
[Liu et al., 2019; Joshi et al., 2020]

Parameter-Efficient Finetuning: Prefix-Tuning, Prompt tuning

Prefix-Tuning adds a **prefix** of parameters, and **freezes all pretrained parameters**.

The prefix is processed by the model just like real words would be.

Advantage: each element of a batch at inference could run a different tuned model.

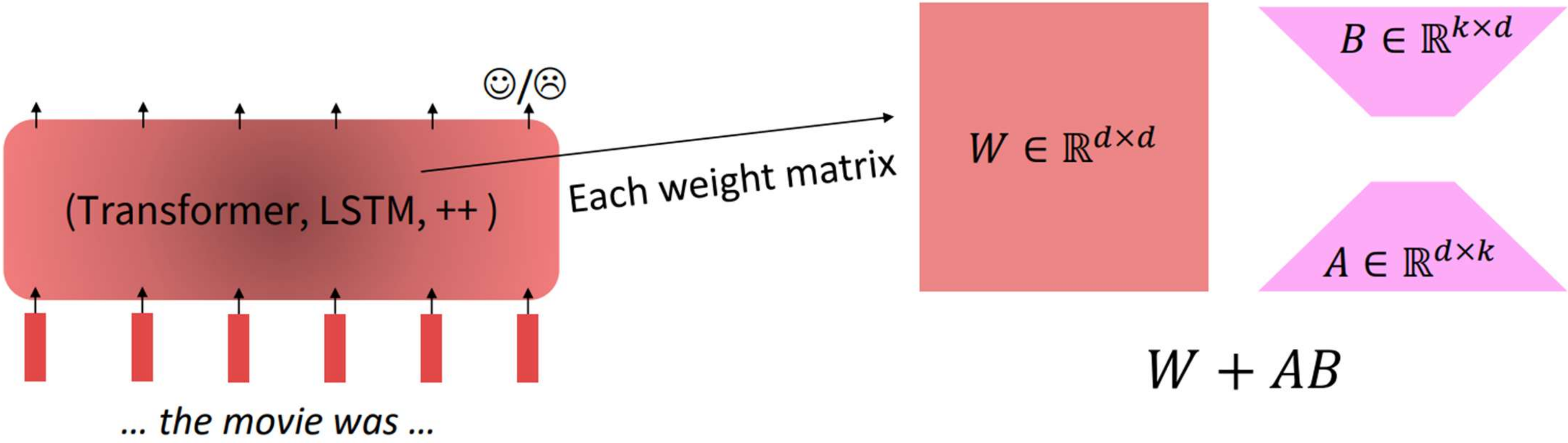


[Li and Liang, 2021; Lester et al., 2021]

Parameter-Efficient Finetuning: Low-Rank Adaptation

Low-Rank Adaptation Learns a low-rank “diff” between the pretrained and finetuned weight matrices.

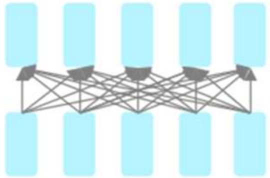
Easier to learn than prefix-tuning.


☺/☹
(Transformer, LSTM, ++)
... the movie was ...
Each weight matrix
$W \in \mathbb{R}^{d \times d}$
$B \in \mathbb{R}^{k \times d}$
$A \in \mathbb{R}^{d \times k}$
$W + AB$

[Hu et al., 2021]

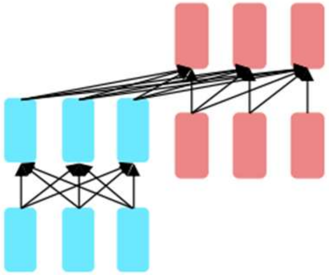
Pretraining for three types of architectures

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



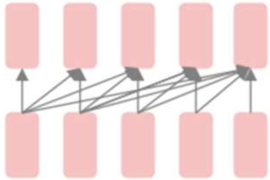
Encoders

- 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?



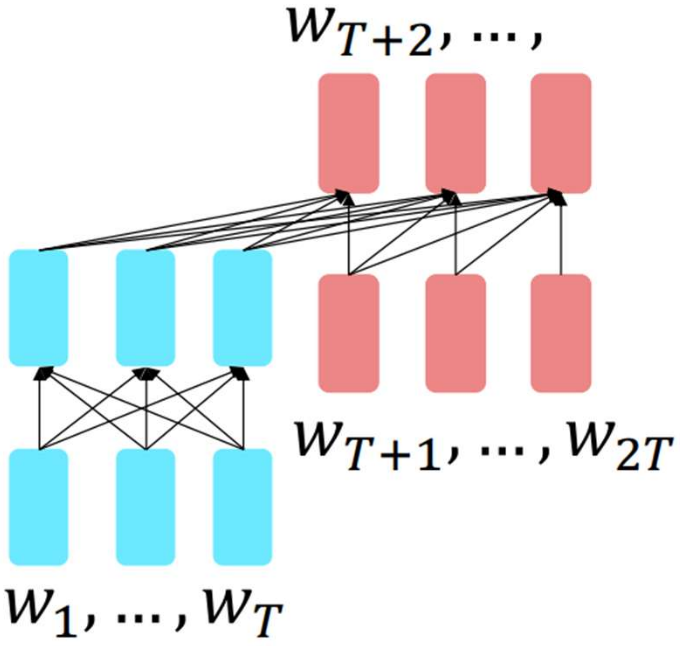
Decoders

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

Pretraining 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.

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



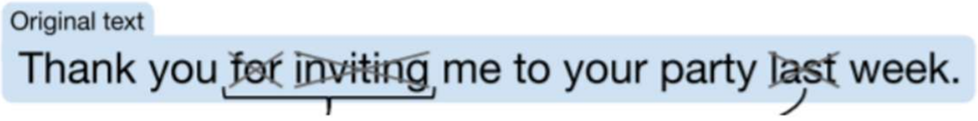
The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.

[Raffel et al., 2018]

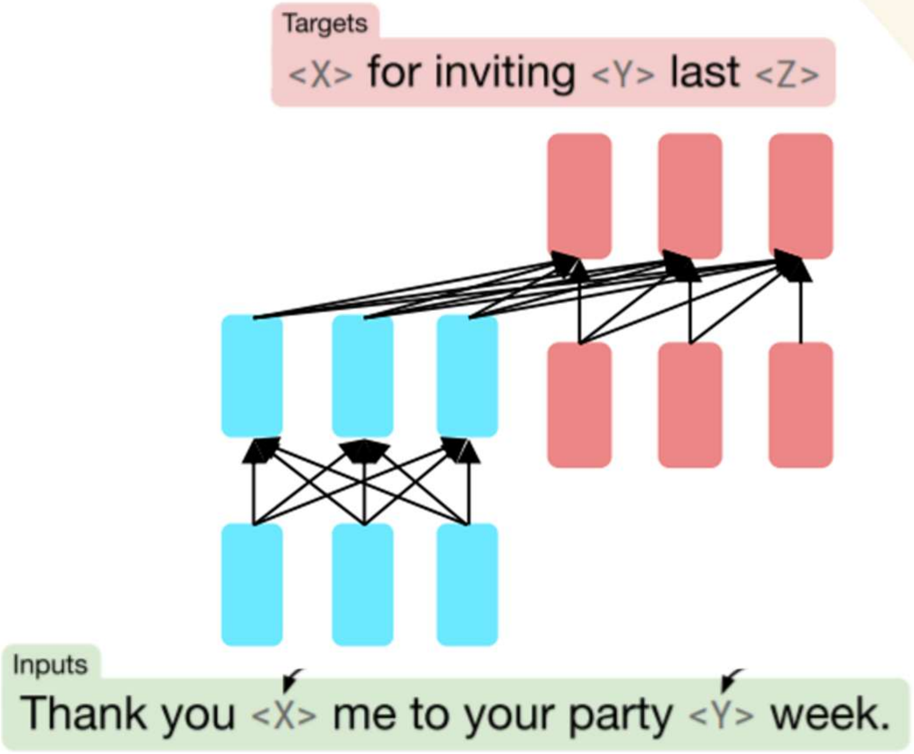
Pretraining encoder-decoders: what pretraining objective to use?

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!

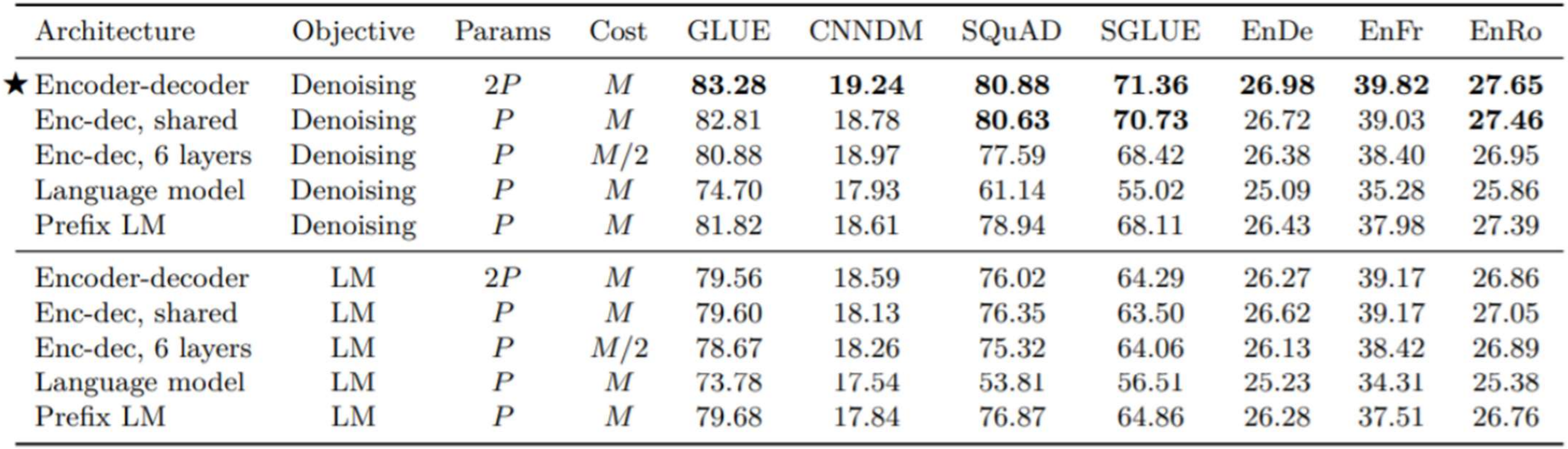


This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.


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

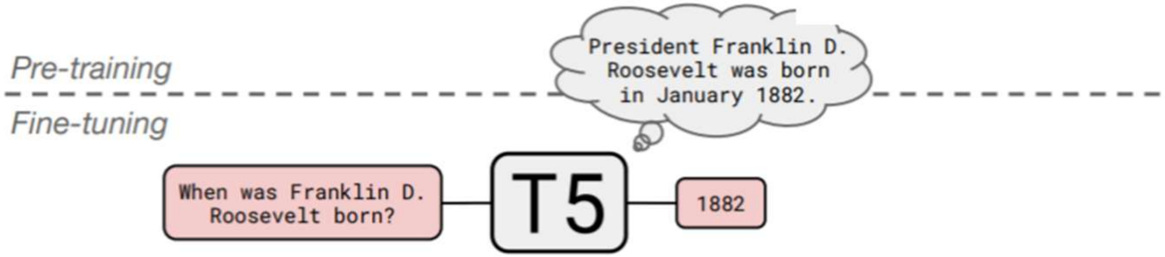


Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88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

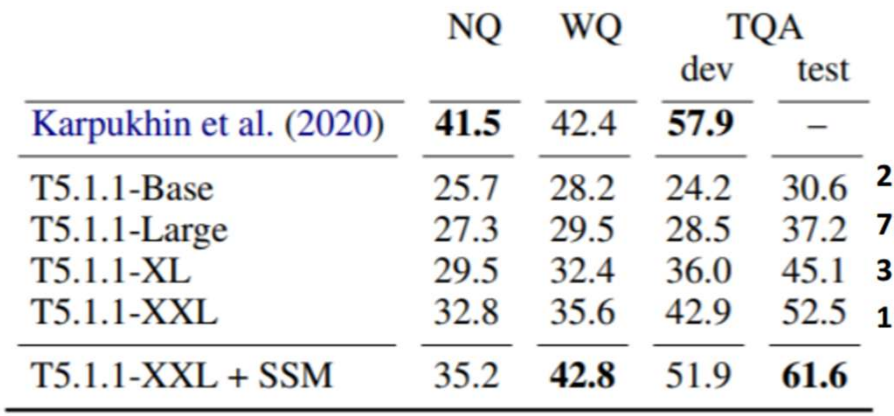
- A fascinating property of T5:

it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.


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

- NQ: Natural Questions
- WQ: WebQuestions
- TQA: Trivia QA

All “open-domain” versions



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

220 million params
770 million params
3 billion params
11 billion params

[Raffel et al., 2018]