

Natural Language Processing with Deep Learning

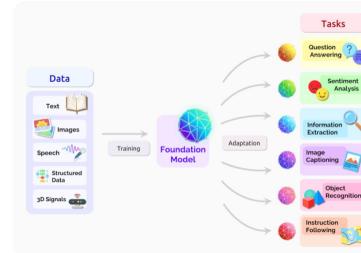
CS224N/Ling284



Tatsunori Hashimoto

Lecture 9: Pretraining

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)	█

5

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.

6

[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.

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

7

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 / *record the record*: the two instances of **record** mean different things.

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

9

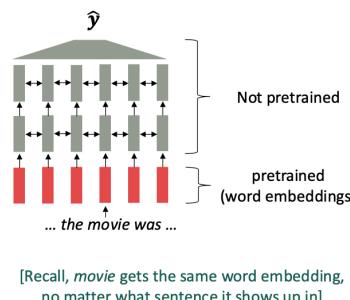
Where we were: pretrained word embeddings

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!



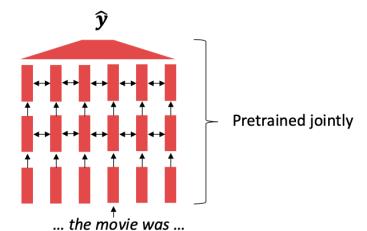
[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

10

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]

11

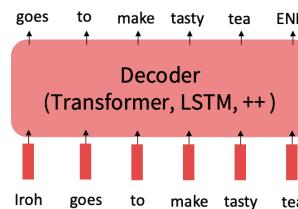
Pretraining through language modeling [Dai and Le, 2015]

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.



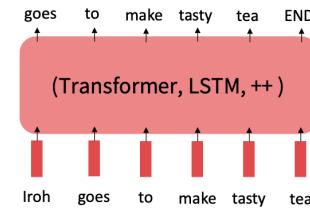
19

The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

Step 1: Pretrain (on language modeling)

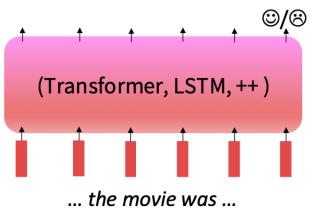
Lots of text; learn general things!



20

Step 2: Finetune (on your task)

Not many labels; adapt to the task!



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 [Rajpurkar+ 2018]	< 50 Million
DCLM-pool [Li+ 2024]	240 Trillion
Estimated 'internet text' [Villalobos 2024]	510T (indexed), 3100T (total)

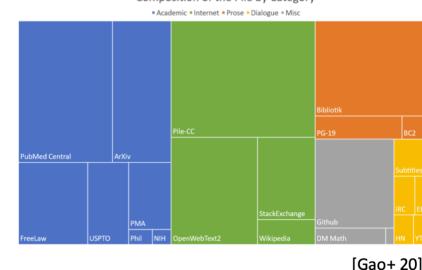
A 10 million times gap in QA to indexed internet

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

Composition of the Pile by Category



[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

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

Reexamining "Fair Use" in the Age of AI

Arts and Humanities, Law, Regulation, and Policy, Machine Learning
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 [v](#) [f](#) [m](#) [in](#) [@](#)



Pretraining encoders: what pretraining objective to use?

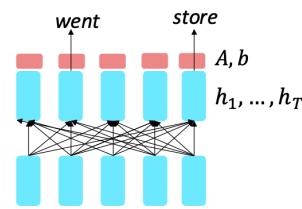
So far, we've looked at language model pretraining. But **encoders get bidirectional context**, so we can't do language modeling!

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**.

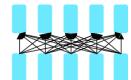


[Devlin et al., 2018]

34

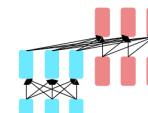
Pretraining for three types of architectures

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



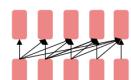
Encoders

- 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

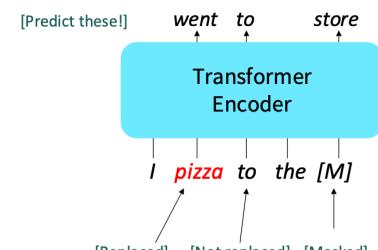
32

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!)

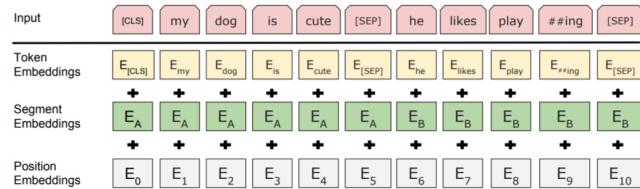


[Devlin et al., 2018]

35

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.

36

[\[Devlin et al., 2018, Liu et al., 2019\]](#)

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.”

37

[\[Devlin et al., 2018\]](#)

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

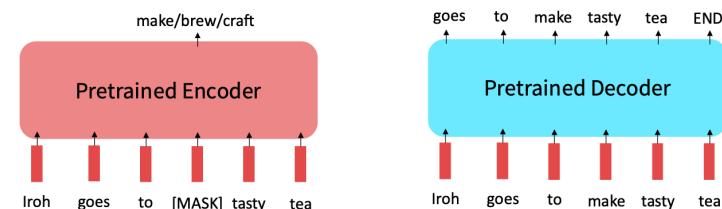
38

[\[Devlin et al., 2018\]](#)

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.



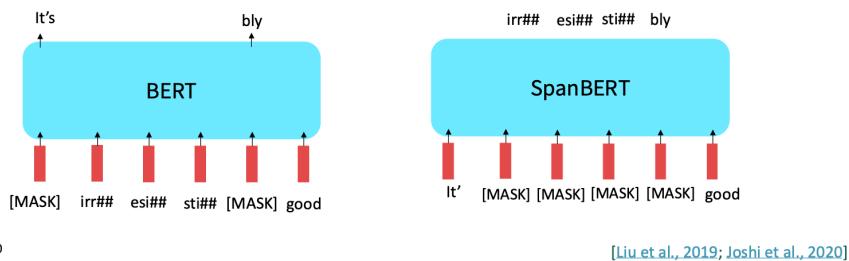
39

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



40

Extensions of BERT

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

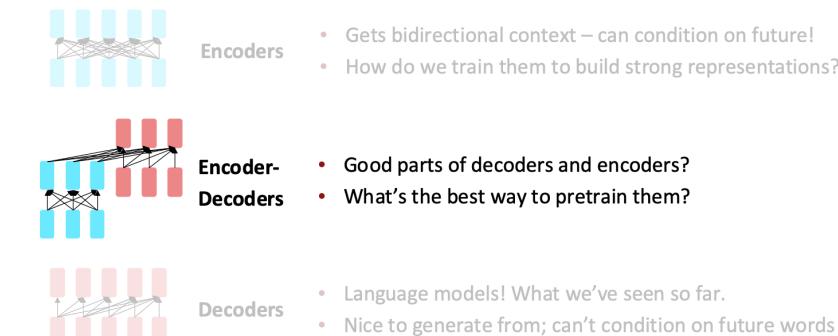
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	94.6/89.4	90.2	96.4
BERT _{LARGE}			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

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

41

Pretraining for three types of architectures

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



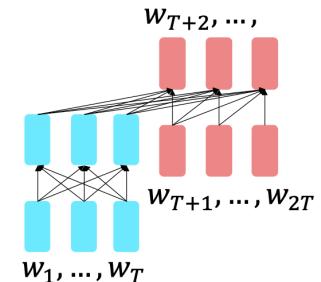
42

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]

43

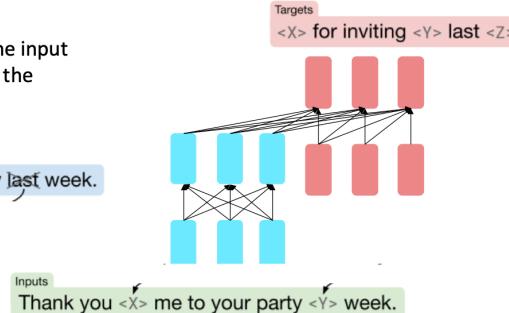
Pretraining encoder-decoders: what pretraining objective to use?

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.



44

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?

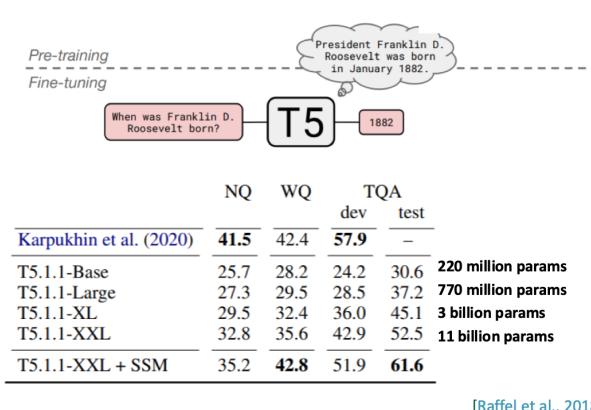
A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.

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	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	

[Raffel et al., 2018]

Pretraining for three types of architectures

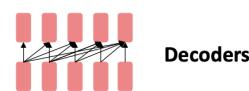
The neural architecture influences the type of pretraining, and natural use cases.



Encoders



Encoder-Decoders



Decoders

- 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 $p(w_t|w_{1:t-1})$.

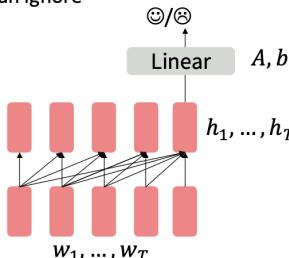
We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$

$$y \sim Ah_T + b$$

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

Gradients backpropagate through the whole network.



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

48

Pretraining decoders

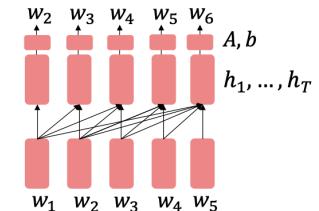
It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!

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)

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$

$$w_t \sim Ah_{t-1} + b$$



[Note how the linear layer has been pretrained.]

Where A, b were pretrained in the language model!

49

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"

50

[Devlin et al., 2018]

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.

51

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

52

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.

54

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 -> "

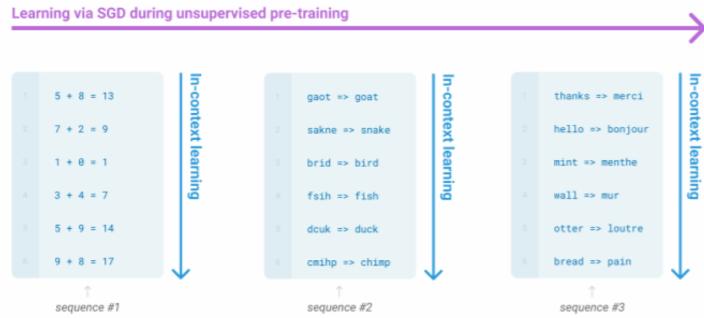
Output (conditional generations):

"loutre..."

55

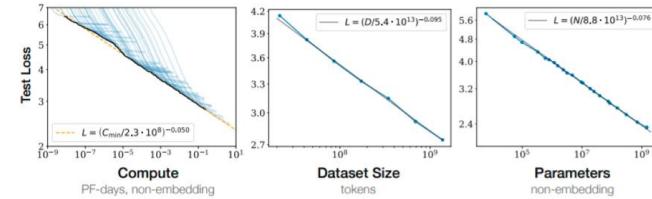
GPT-3, In-context learning, and very large models

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



56

Why scale? Scaling laws



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

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!

62