

## 01. BERT

Why was BERT way ahead of its time?

- 
- 
- 

Because it was a masked language model even during pre-covid days!



CS 2731 / ISSP 2230

# Introduction to Natural Language Processing

Session 14: Transformers part 2, pretraining, GPT and BERT

---

Michael Miller Yoder

February 26, 2024

# Course logistics

- Project proposal presentations **next Mon during class**
  - Aim for **~5 min** presentations
  - There will be Q+A after each presentation
  - Add your slides here:  
[https://docs.google.com/presentation/d/1Xu2ebscCVlKYe\\_A1orOZ0Q\\_EiSbZTJyjAvgVpPffUeE4/edit?usp=sharing](https://docs.google.com/presentation/d/1Xu2ebscCVlKYe_A1orOZ0Q_EiSbZTJyjAvgVpPffUeE4/edit?usp=sharing)
  - Instructions are on the [project website](#) and in the slide deck
- Project proposal feedback this week
- [Homework 3](#) is **due Thu Mar 7**
- **This Wed** will be a BERT/LLM lab and discussion day
  - Bring a laptop

# Lecture overview: Transformers part 2, pretraining, BERT & GPT

- Pretraining and finetuning
- Pretraining 3 ways
  - Encoders (BERT)
  - Encoder-decoders (T5)
  - Decoders (GPT)
- Contextual word embeddings

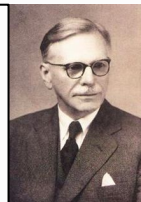
# Pretraining and finetuning

---

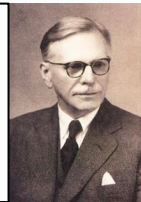
# Word meaning and context

Summary of **distributional semantics** and motivation for **word2vec**:

"You shall know a word by the company it keeps" [Firth 1957]



"the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously"  
[Firth 1935]



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

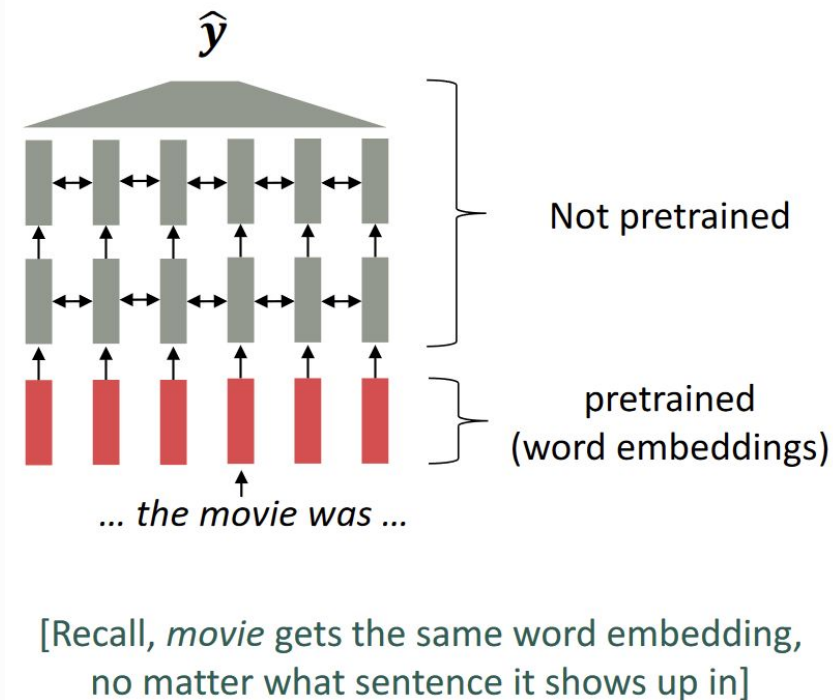
# Where we are: pretrained word embeddings

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:

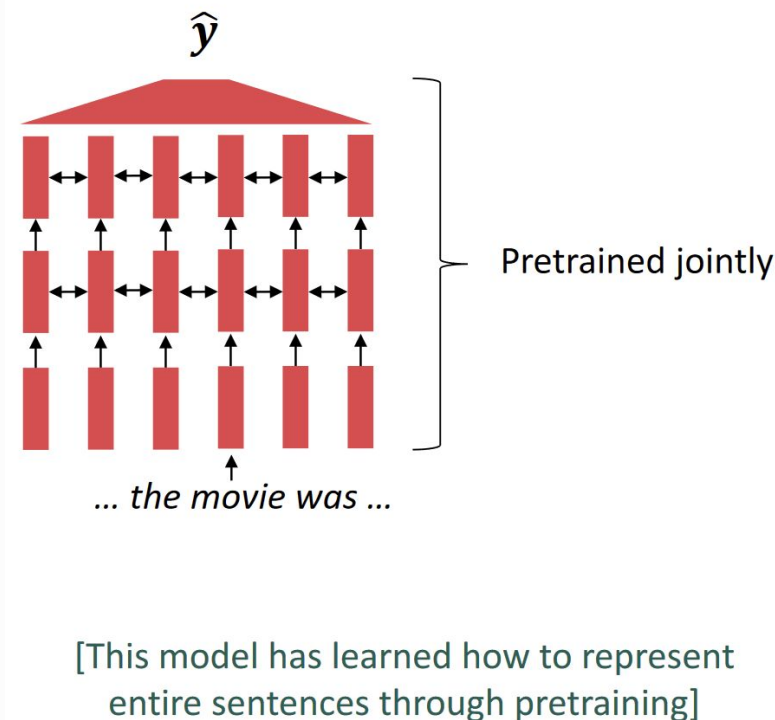
- The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network are randomly initialized!



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

In contemporary 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





# What can we learn from reconstructing the input?

- MIT is located in \_\_\_\_\_, Massachusetts.
- I put \_\_\_ fork down on the table.
- The woman walked across the street, checking for traffic over \_\_\_ shoulder.
- I went to the ocean to see the fish, turtles, seals, and \_\_\_\_\_.
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was \_\_\_\_\_.
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the \_\_\_\_\_.
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, \_\_\_\_\_

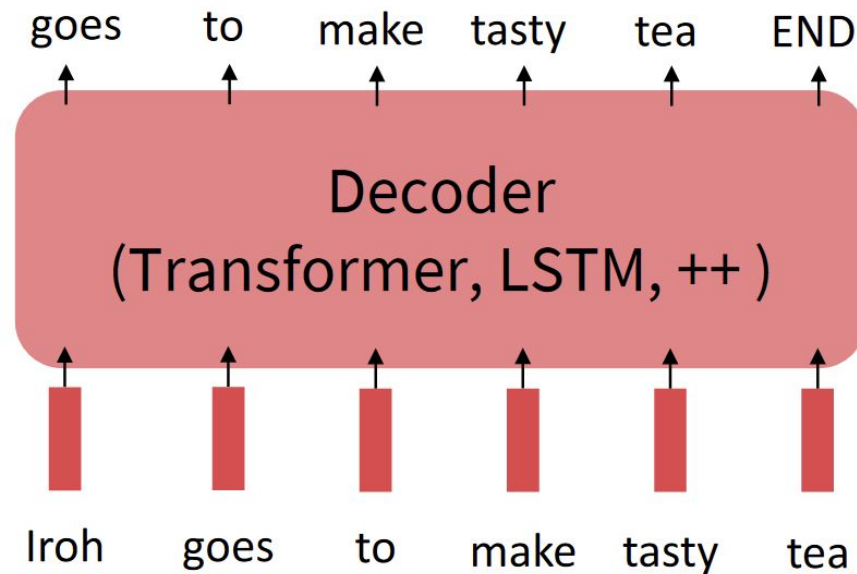
# Pretraining through language modeling [Dai and Le 2015]

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.

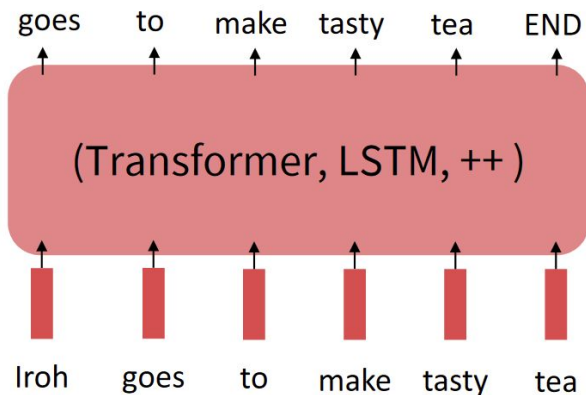


# The pretraining + finetuning paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

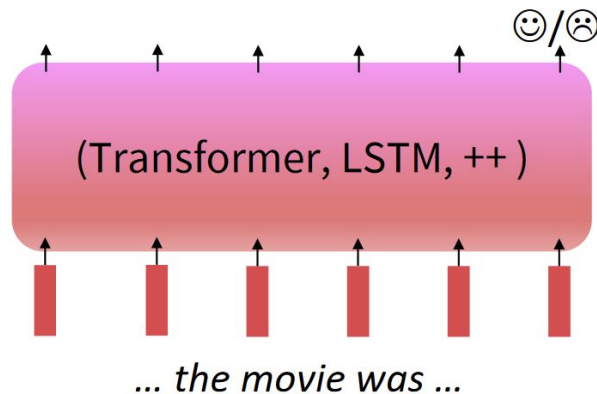
## Step 1: Pretrain (on language modeling)

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!

# A reminder: Byte Pair Encoding (BPE, Sennrich+ 2016)

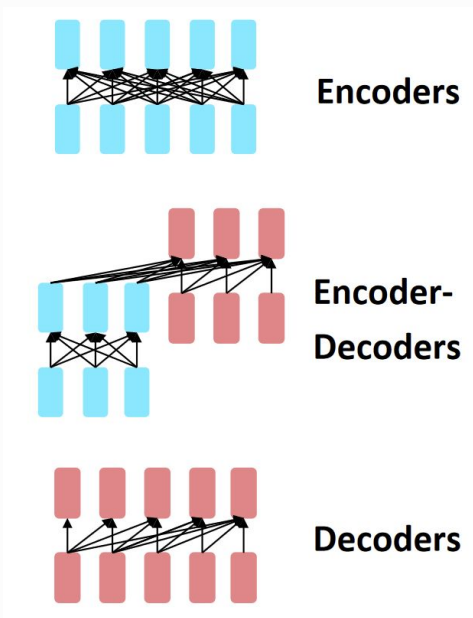
- LLMs generally use **subword tokenization**
- Merges frequently seen sequences of characters together into tokens
- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
  - Until  $k$  merges have been done.
- Allows them to generalize to unseen words, handle misspelling, novel words

Model pretraining 3 ways:  
encoders, encoder-decoders, decoders

---

# Pretraining for 3 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?
- Nice to generate from; can't condition on future words

# Pretraining encoders (BERT)

---



# Pretraining encoders: what pretraining objective to use?

- So far, we've looked at language model pretraining.
- But encoders can access to bidirectional context.
- Let's use it in training!
- BERT is pretrained with 2 objectives
  - Masked language modeling
  - Next sentence prediction

# The Cloze Task

- The **cloze** task comes from psycholinguistics (the branch of linguistics and cognitive science that uses experimental methods to study how language works in human brains).
- It is a fill-in-the-blank task:

**He drove the yellow \_\_\_\_\_ into the front of our house.**

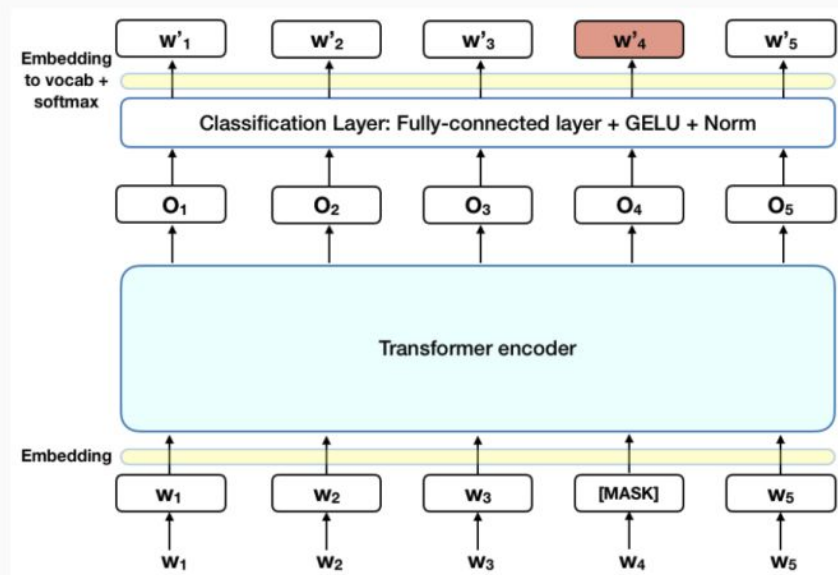
- Subjects are presented with these frames and asked to fill in the missing words
- This allows experimenters to assess what a speaker understands about grammar, semantics, etc.
- According to the original BERT paper, this task provided the inspiration for BERT's masked language modeling (MLM) training task.
- But compare various kinds of denoising algorithms.

# BERT is Trained to Perform Masked Language Modeling

- Since, in BERT's innards, everything is literally connected to everything, each word would **indirectly see itself**
- Enter masking!
- In training, random word are concealed from BERT using the **[MASK]** token
- BERT is trained to guess these masked words
- Take the sentence: “the girl was playing outside.”
  - Suppose that a **[MASK]** token is inserted in place of *outside*, thus masking it
  - We then have “the girl was playing **[MASK]**.”
  - The model now cannot “see” *outside*; instead it is trained to generate it based on context
  - This is part of why BERT can be so good at representing words according to the context in which they occur (not just the distribution of identically-spelled tokens)

# How BERT Is Trained to Perform Masked Language Modeling

1. Add classification layer
2. Multiply output vectors by embedding layer  $\rightarrow$  vocabulary dimension
3. Calculate probabilities using softmax

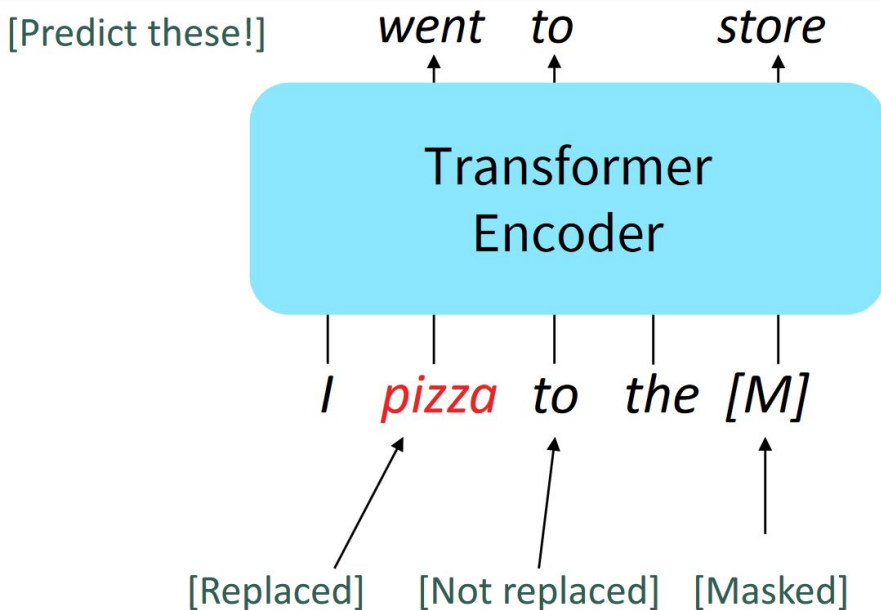


# BERT: Bidirectional Encoder Representations from Transformers

Devlin et al. 2018 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!)

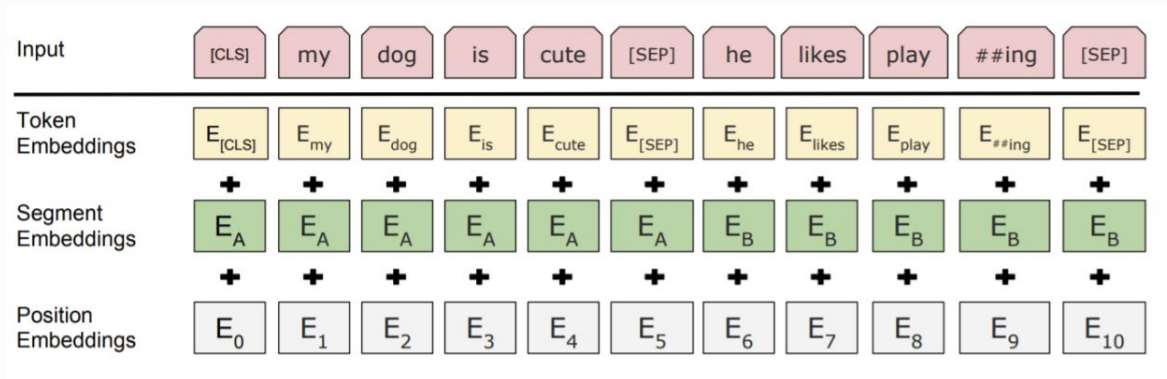


# BERT is also Trained to Perform Next Sentence Prediction

- Suppose you are given two pairs of sentences:
  1. Good pair
    - 1.1 Pickles are delicious.
    - 1.2 However, cucumbers make me burp.
  2. Bad pair
    - 2.1 Pickles are delicious.
    - 2.2 The NASDAC was up 10 points today amid heavy trading.
- For each pair, you (a human) can tell whether the first and the second are likely to occur in sequence
- BERT is trained to do the same
- As a result, BERT learns patterns that exist above the level of the sentence

# BERT: Bidirectional Encoder Representations from Transformers

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



BERT was also trained to predict whether one chunk follows the other or is randomly sampled (**next sentence prediction**). Later work has argued this “next sentence prediction” is not necessary [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.”



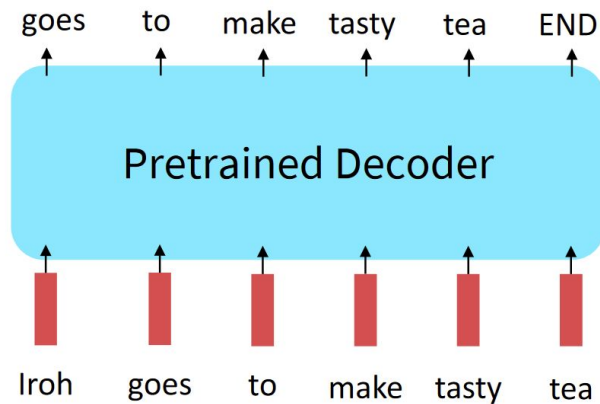
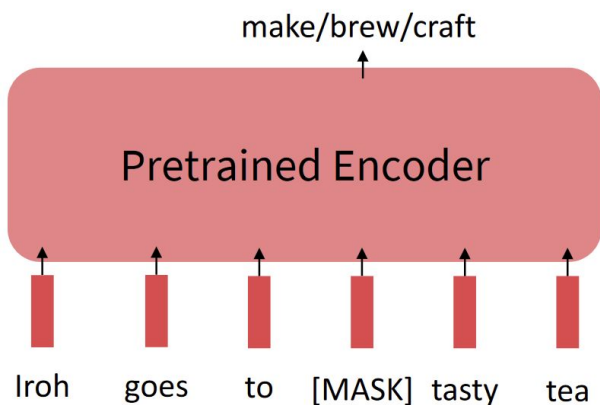
# BERT: Bidirectional Encoder Representations from Transformers

- 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>

# BERT: Bidirectional Encoder Representations from Transformers

- Those results looked great! Why not use 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.

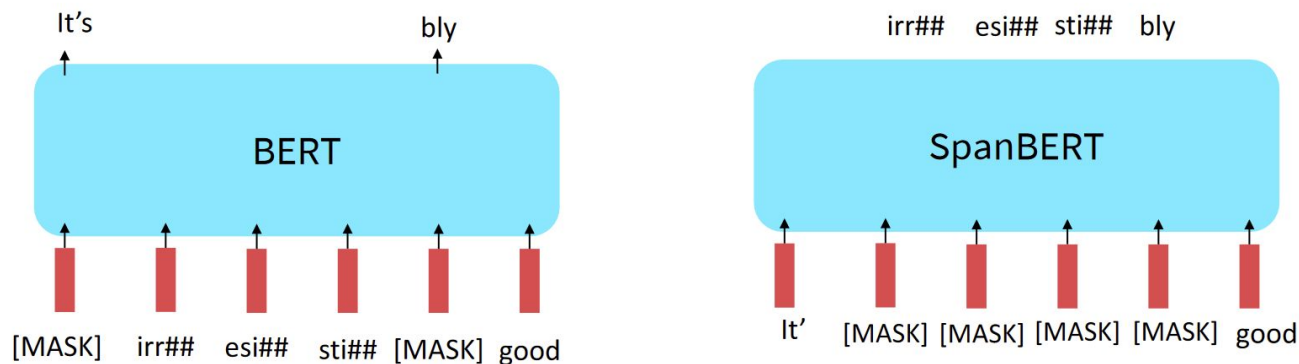


# Extensions of BERT

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

Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa [Liu et al. 2019]: mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT [Joshi et al. 2020]: masking contiguous spans of words makes a harder, more useful pretraining task



# Pretraining encoder-decoders (T5)

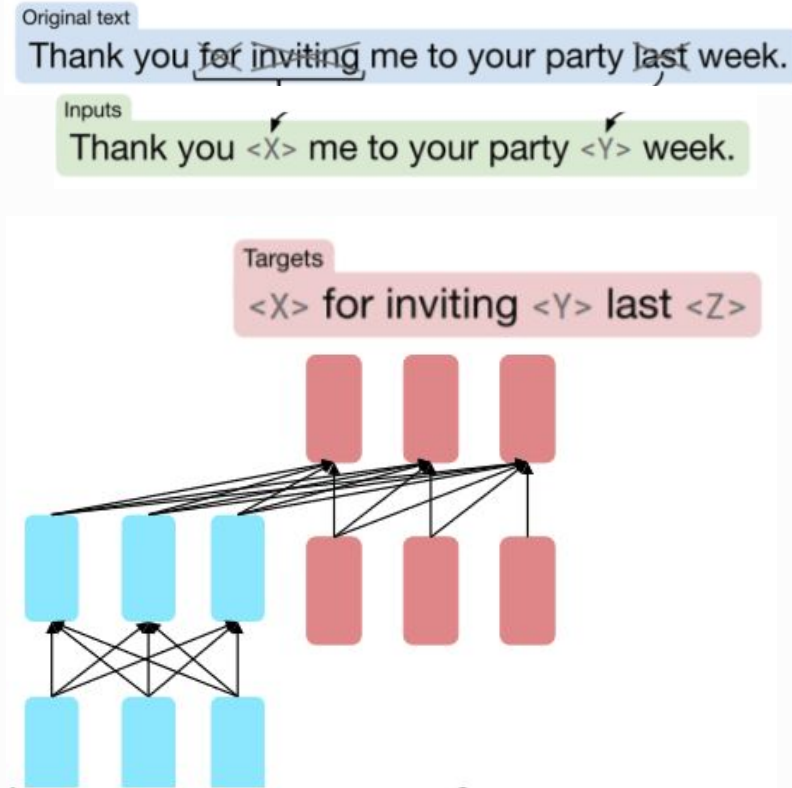
---

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

What Raffel et al. 2020 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



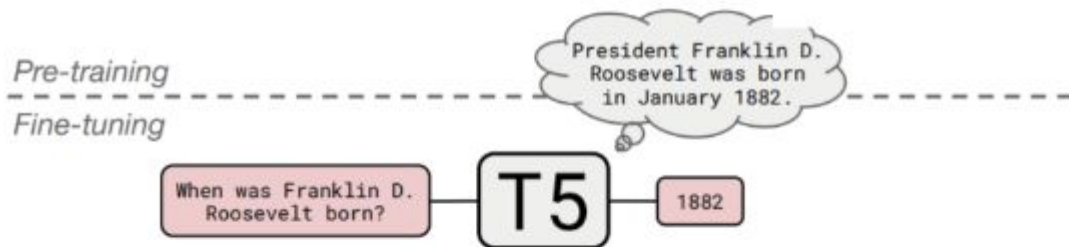
# Finetuned T5

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



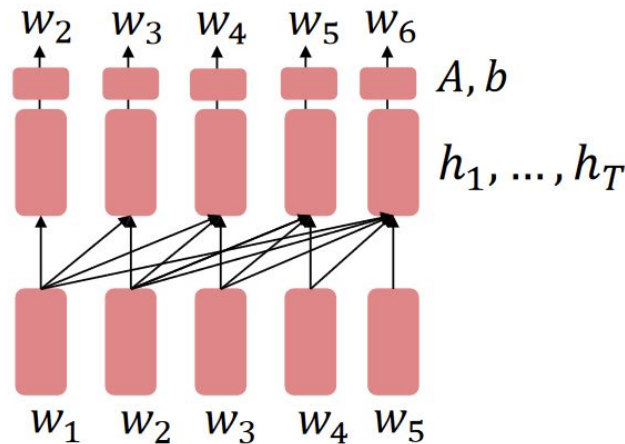
	NQ	WQ	TQA		
			dev	test	
<a href="#">Karpukhin et al. (2020)</a>	<b>41.5</b>	42.4	<b>57.9</b>	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	<b>220 million params</b>
T5.1.1-Large	27.3	29.5	28.5	37.2	<b>770 million params</b>
T5.1.1-XL	29.5	32.4	36.0	45.1	<b>3 billion params</b>
T5.1.1-XXL	32.8	35.6	42.9	52.5	<b>11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

# Pretraining decoders (GPT)

---

# Pretrained decoders for generation

- 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)
- $A, b$  were pretrained in the language model



[Note how the linear layer has been pretrained.]

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

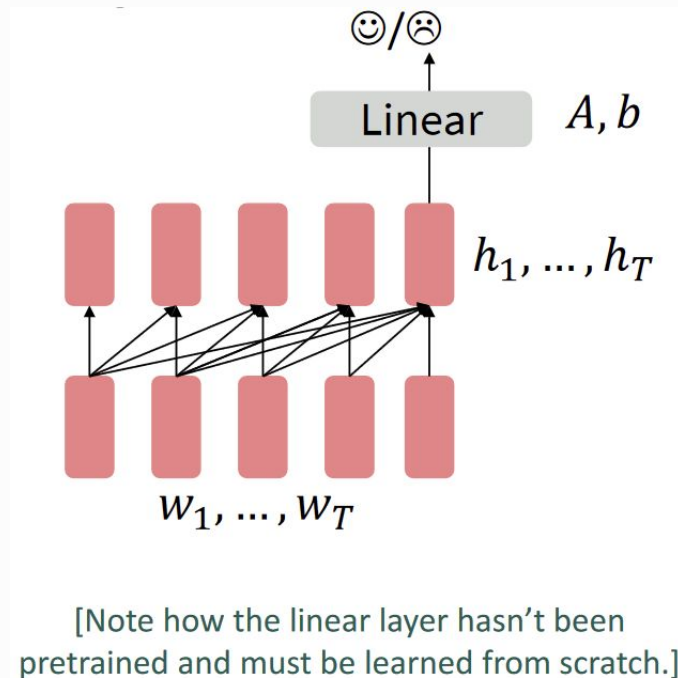


# Pretrained decoders for classification

- When using language model pretrained decoders for classification, 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.



# 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* } **entailment.**  
Hypothesis: *The person is near the door*

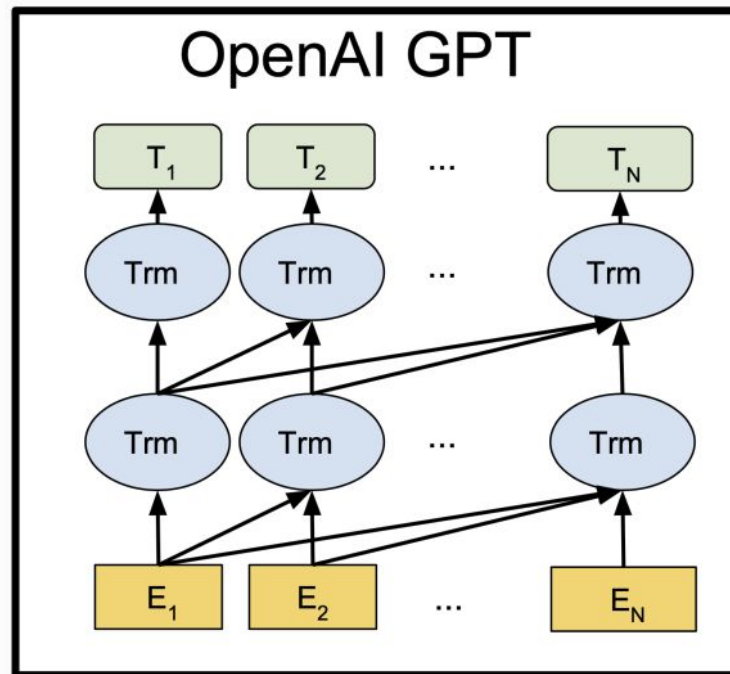
Can format the input as a sequence of tokens for the decoder.

[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.

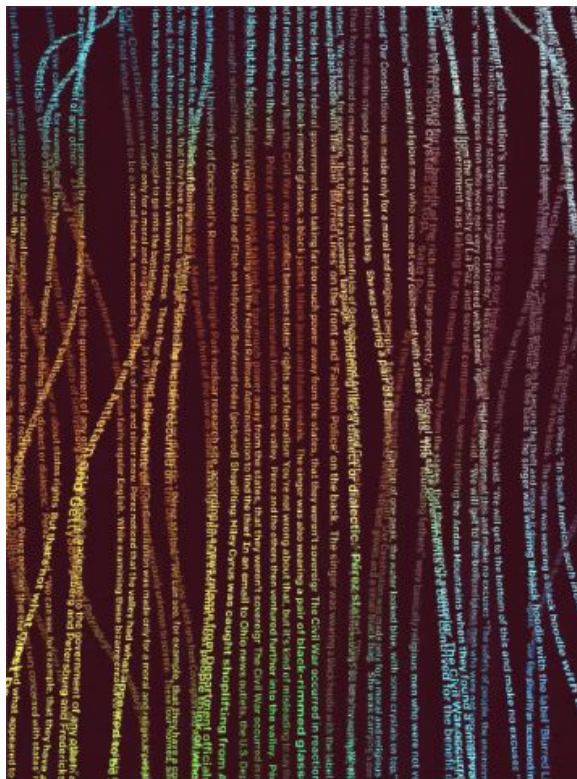
# Generative Pretrained Transformer (GPT; Radford et al. 2018)

- 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”



# GPT-2, GPT-3, GPT-4 from OpenAI

- They are basically larger and larger autoregressive transformer LMs trained on larger and larger amounts of data
- They have shown amazing language generation capability when you give it a prompt (aka. prefix, the beginning of a paragraph)



# Generation example from the GPT-2 model

SYSTEM PROMPT  
(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.*

MODEL COMPLETION  
(MACHINE-WRITTEN,  
10 TRIES)

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.

A sample from GPT2 (with top-k sampling)

# 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.

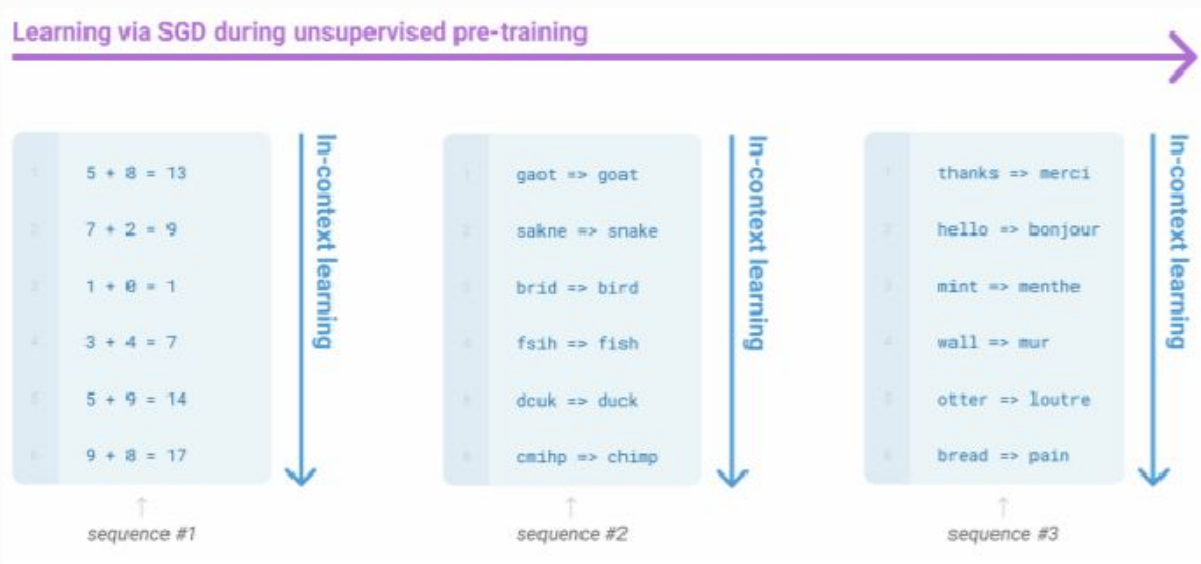
Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

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 steps seem to specify the task to be performed.



# 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.

- MIT is located in \_\_\_\_\_, Massachusetts. [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 Fibonacci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.



# Contextual word embeddings

---

# The meaning of words is contextual

Static word embeddings (word2vec, GloVe, etc):

"You shall know a word by the company it keeps" [Firth 1957]



"the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously"  
[Firth 1935]



Let's use LLMs like BERT to get contextual word and sentence embeddings!

# Static Word Embeddings Ignore Homography and Polysemy

Suppose you retrieve the embeddings for the following two sentences from a set of static word embeddings like GloVe or Word2Vec (trained via skip-gram or CBOW):

1. Lifting Dell laptops causes lower **back** pain.
2. He went **back** to his office to sulk.
3. She will **back** her truck into a fire hydrant.

The meanings of these three words are rather different but their embeddings would be the same. **This is problematic.**

# Contextual embeddings give words representations based on context

If you fed the same sentences...

1. You caused me lower **back** pain.
2. He went **back** to his office to sulk.

...to a contextual model like BERT or ELMo [Peters et al. 2018], the two words would have different embeddings (reflecting their differing meanings).













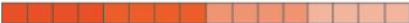
ELMo was a model that provided contextual embeddings based on bidirectional LSTMs, not transformers like BERT

# How Do You Get Embeddings from BERT?

- BERT<sub>BASE</sub> has 12 layers
- The output of each base is an embedding
- Choose one of these, or some combination

# To concatenate or sum?

What is the best contextualized embedding for “Help” in that context?  
For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
12 	First Layer 	91.0
• • •	Last Hidden Layer 	94.9
7 	Sum All 12 Layers	95.5
6 		
5 		
4 		
3 	Second-to-Last Hidden Layer 	95.6
2 	Sum Last Four Hidden	95.9
1 		
		
Help		
	Concat Last Four Hidden 	96.1

*Questions?*