01. BERT

Why was BERT way ahead of its time?

-
-
-

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

Why are they known are transformers?

Because they're networks in disguise!

CS 2731 Introduction to Natural Language Processing

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

Michael Miller Yoder

October 16, 2023

University of **PITTSBURGH** | **School of Computing and Information**

Course logistics

- Homework 2 grades out by the end of the week
- Project proposal presentations **this Wed during class**
  - **4 min** presentations + **2 min** questions (6 min total) for each group
  - Add your slides here:
    https://docs.google.com/presentation/d/1x79rdTho1_NAEEjZbiX-ehwJ36DEn_BatmOO4rDpDVQ/edit?usp=sharing
  - Instructions are on the project website and in the slide deck

Midterm OMETs: thank you for the feedback!

- ~40% participation rate
- What has helped your learning
  - Reading quizzes
  - Slides
  - Demo on Colab
- What has made learning difficult
  - Need to know ML concepts
  - Quick time constraint on homeworks
- Changes to help you learn
  - Programming examples/lab classes
- Planned changes
  - More in-class code examples/labs
  - More pauses to catch up in class
  - More time for homeworks

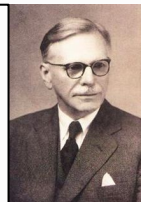# Lecture overview: Transformers part 2, pretraining, BERT & GPT

- Pretraining and finetuning

- Pretraining 3 ways

  - Encoders (BERT)

  - Encoder-decoders (T5)

  - Decoders (GPT)

- Contextual word embeddings
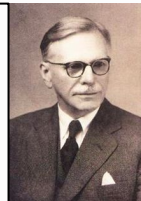
Pretraining

Motivating 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]

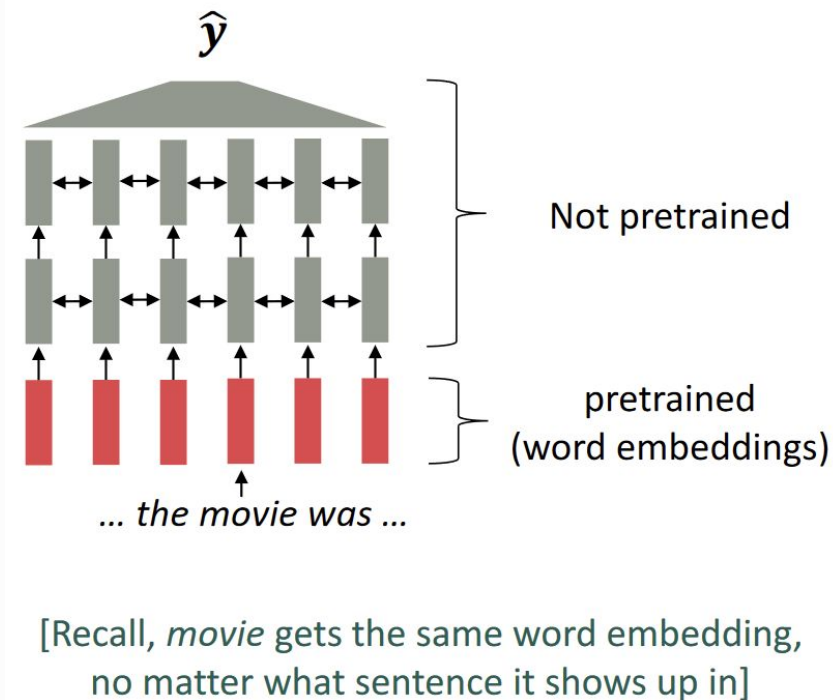Consider *I record the record*: the two instances of *record* mean different things.

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.
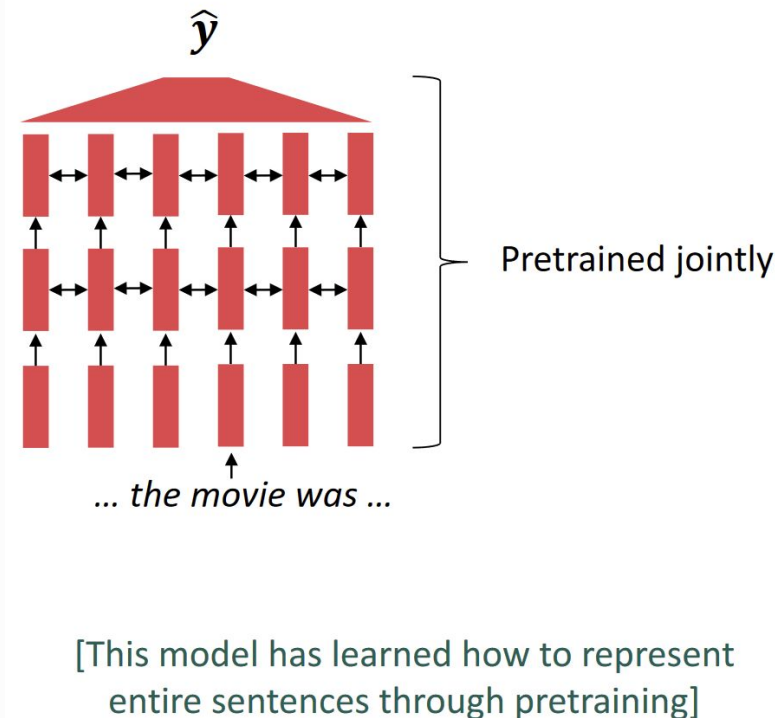
Some issues to think about:

· 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!

$\widehat{y}$

Not pretrained

pretrained (word embeddings)

... *the movie was* ...

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

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

$$\hat{\boldsymbol{y}}$$

Pretrained jointly

... *the movie was* ...

[This model has learned how to represent entire sentences through pretraining]

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.
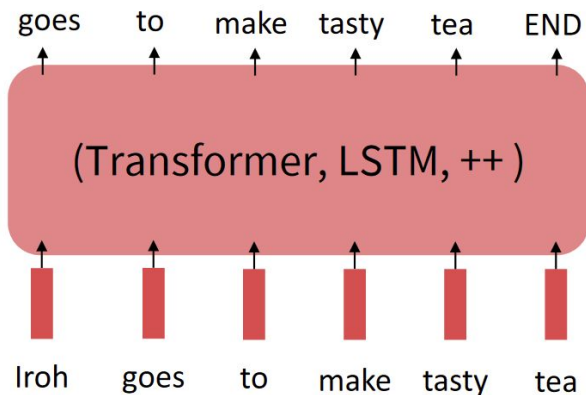
· Save the network parameters.
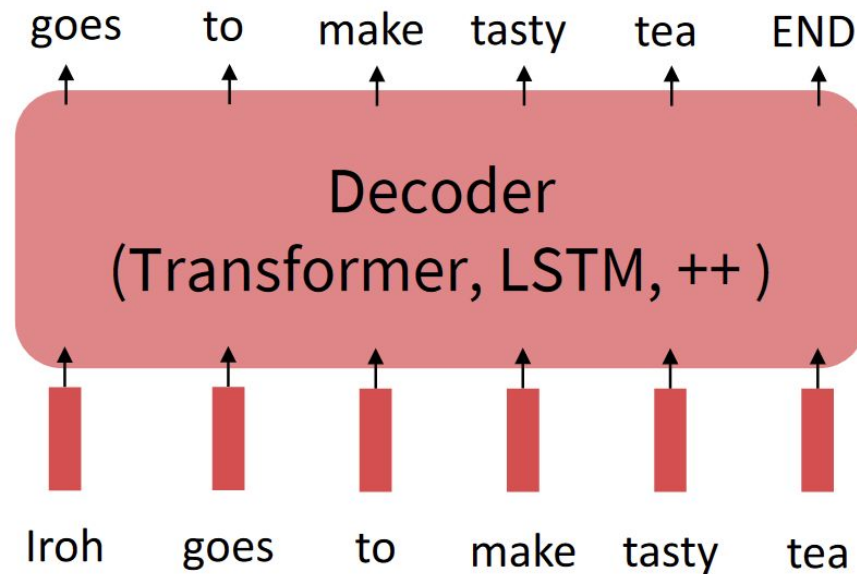
The pretraining + finetuning paradigm

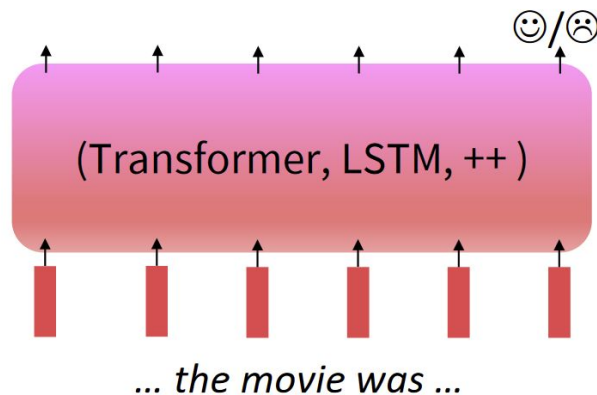Pretraining can improve NLP applications by serving as parameter initialization.

Step 1: Pretrain (on language modeling)
Lots of text; learn general things!

goes   to   make   tasty   tea   END

(Transformer, LSTM, ++)

Iroh   goes   to   make   tasty   tea

Step 2: Finetune (on your task)
Not many labels; adapt to the task!

☺/☹

(Transformer, LSTM, ++)

... the movie was ...

12

# 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!

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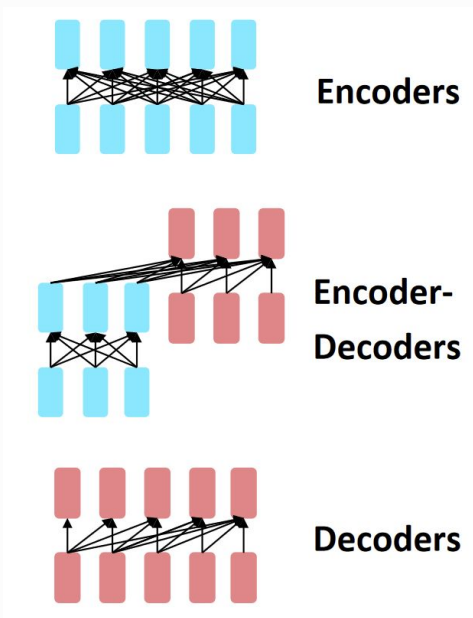
Encoders
- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?

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

Slide adapted from John Hewitt

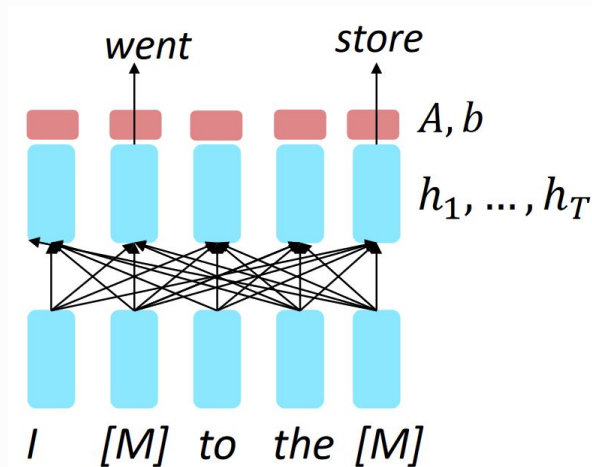15

Pretraining encoders (BERT)

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.

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

Only add loss terms from words that are "masked out." Called **Masked LM**.

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

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 → vocabulary dimension
3. Calculate probabilities using softmax

Slide credit: David Mortensen
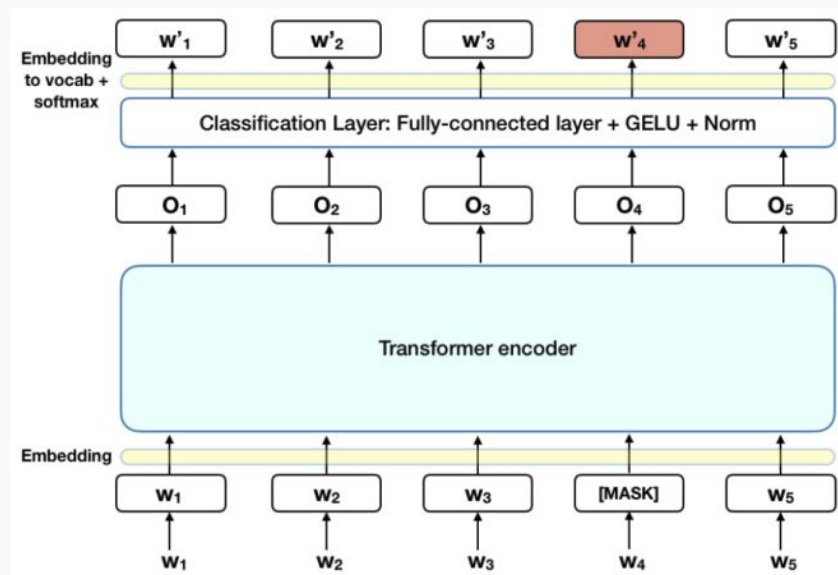
Devlin et al. 2018 proposed the "Masked LM" objective and released the weights of a pretrained tansformer, a model they labeled BERT.

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!)
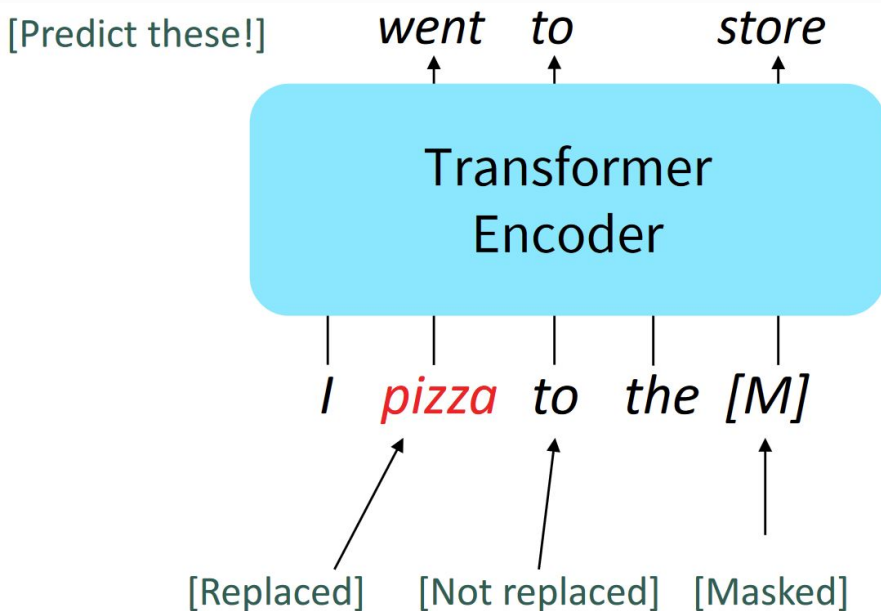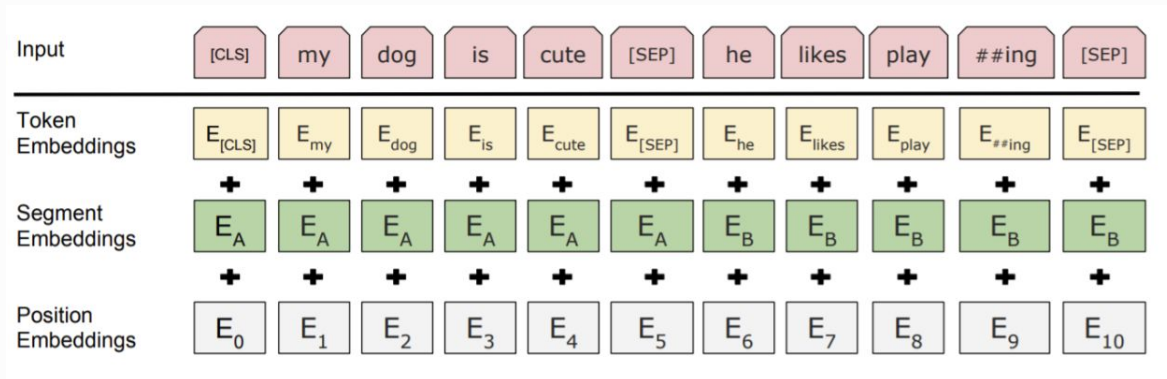
BERT is also Trained to Perform Next Sentence Prediction

- 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:

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 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."

Slide adapted from John Hewitt

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)	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
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**

Slide adapted from John Hewitt

BERT: Bidirectional Encoder Representations from Transformers

- 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.
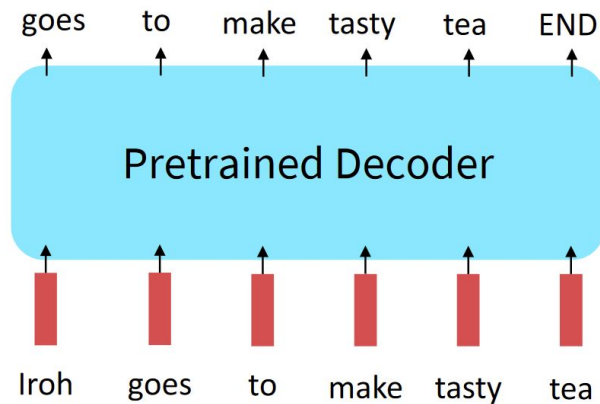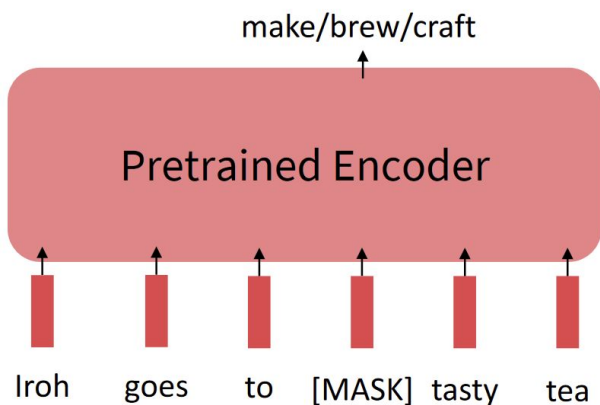
Extensions of BERT

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

Slide adapted from John Hewitt
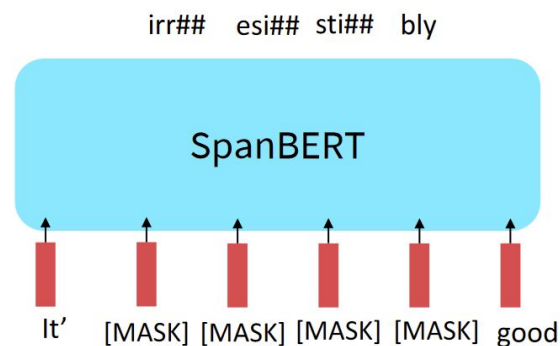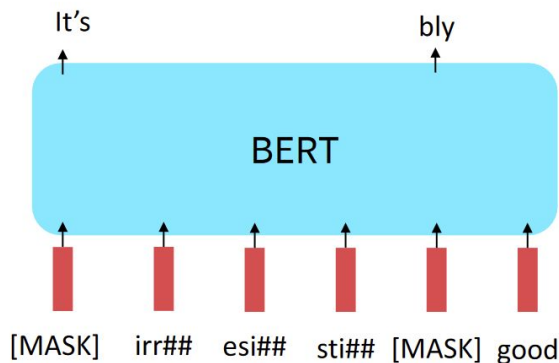
Pretraining encoder-decoders (T5)

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
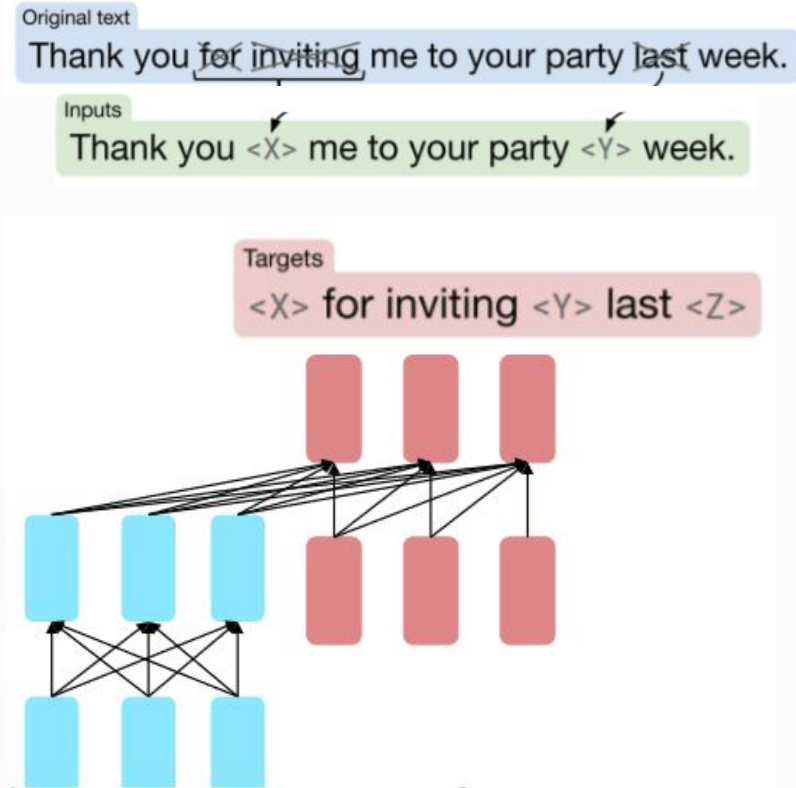
Finetuned T5

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

NQ: Natural Questions

WQ: WebQuestions

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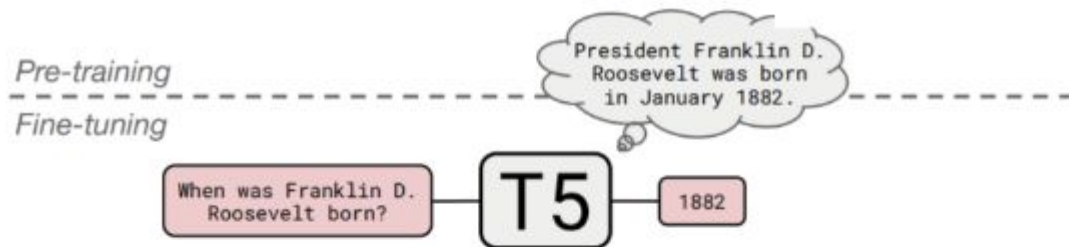
	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL +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

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

$w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6$

A, b

$h_1, \dots, h_T$

$w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$
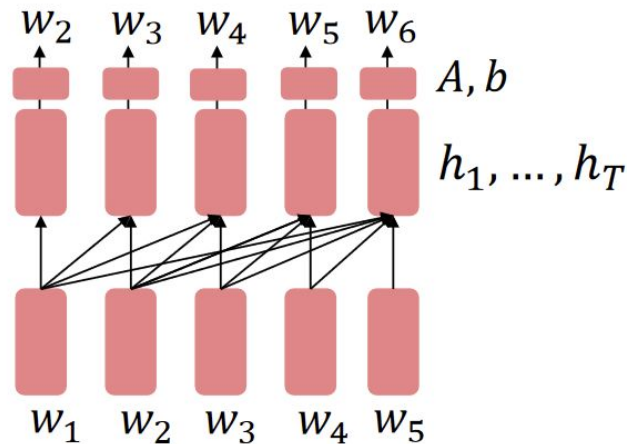
[Note how the linear layer has been pretrained.]

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

Slide adapted from John Hewitt

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.

😊/☹️

Linear $\quad A, b$

$h_1, \dots, h_T$

$w_1, \dots, w_T$

[Note how the linear layer hasn't been pretrained and must be learned from scratch.]
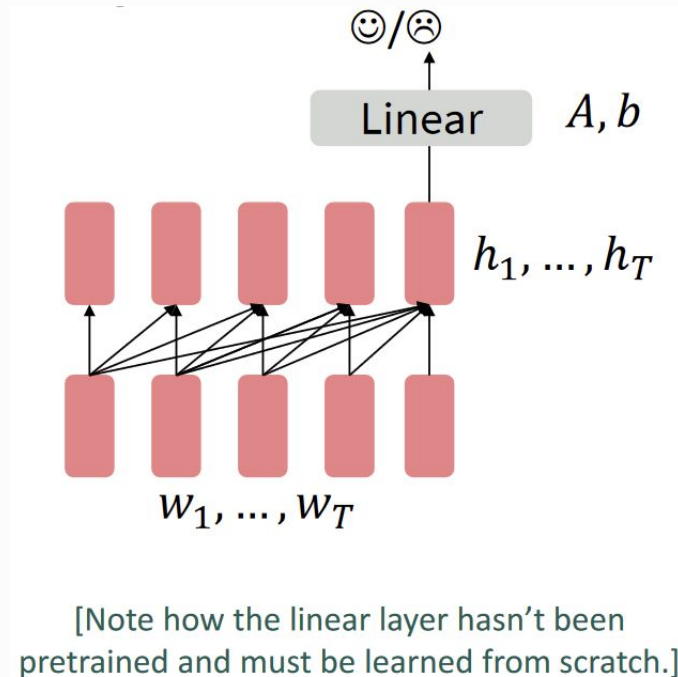
How do we format inputs to our decoder for finetuning tasks?

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.

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"

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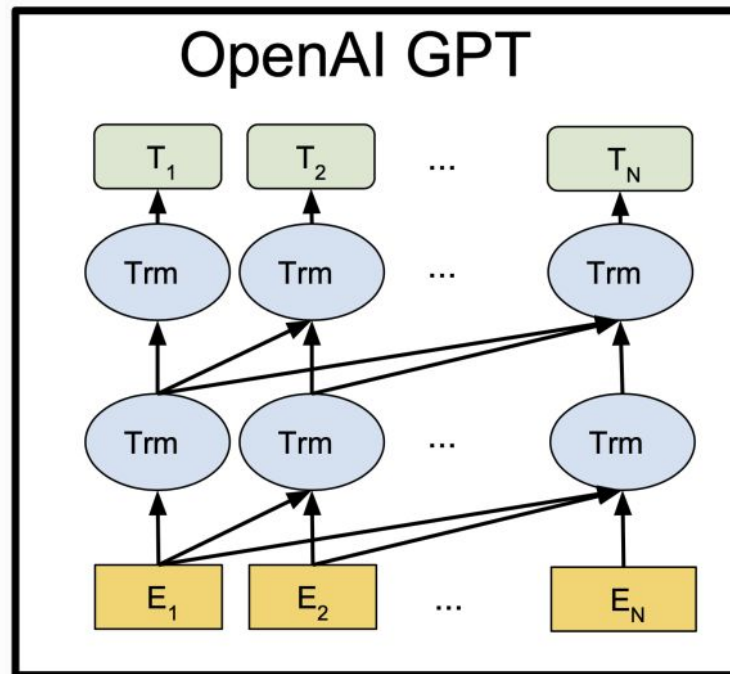
GPT-2, GPT-3, GPT-4 from OpenAI

- They are basically larger and larger autoregressive transformer LM 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)
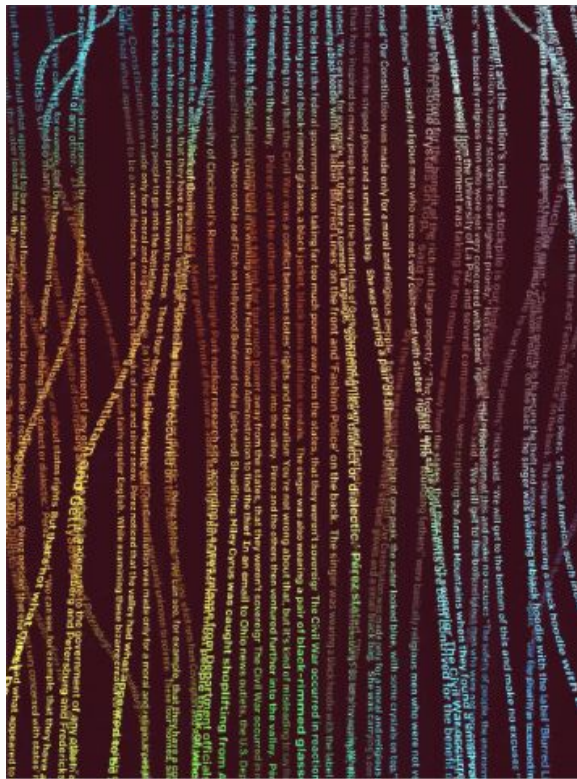
Generation example from the GPT-2 model

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)

The top-k sampling algorithm

We will represent $P(\cdot | W_{1..i})$ by $p = (p_1, p_2, \ldots, p_{|V|})$, where the elements is sorted that $p_1 \geq p_2 \geq p_3 \ldots \geq p_{|V|}$.

Top-K sampling transforms p to $\hat{p}$ by:

$$\widehat{p_i} = \frac{p_i \cdot 1\{i \leq K\}}{Z}$$

And we sample W_{i+1} from $\hat{p}$.

$$\sum_{w \in V_{\text{top-K}}} P(w | \text{"The"}, \text{"car"}) = 0.99$$

$P(w | \text{"The"}, \text{"car"})$

drives is turns stops down a not the small told
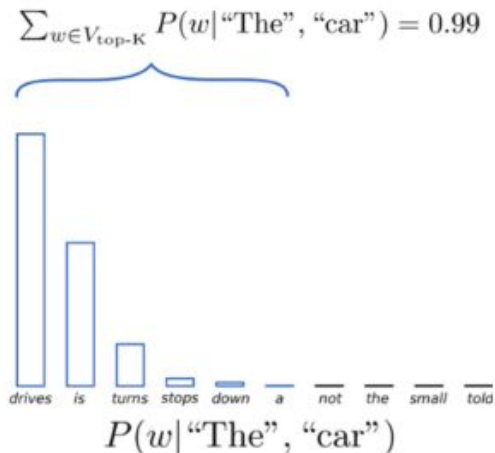
Lower *k* provides higher quality, but less diversity

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**.

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.

Learning via SGD during unsupervised pre-training

sequence #1
5 + 8 = 13
7 + 2 = 9
1 + 0 = 1
3 + 4 = 7
5 + 9 = 14
9 + 8 = 17
In-context learning

sequence #2
gaot => goat
sakne => snake
brid => bird
fsih => fish
dcuk => duck
cmihp => chimp
In-context learning

sequence #3
thanks => merci
hello => bonjour
mint => menthe
wall => mur
otter => loutre
bread => pain
In-context learning

Slide adapted from John Hewitt

41
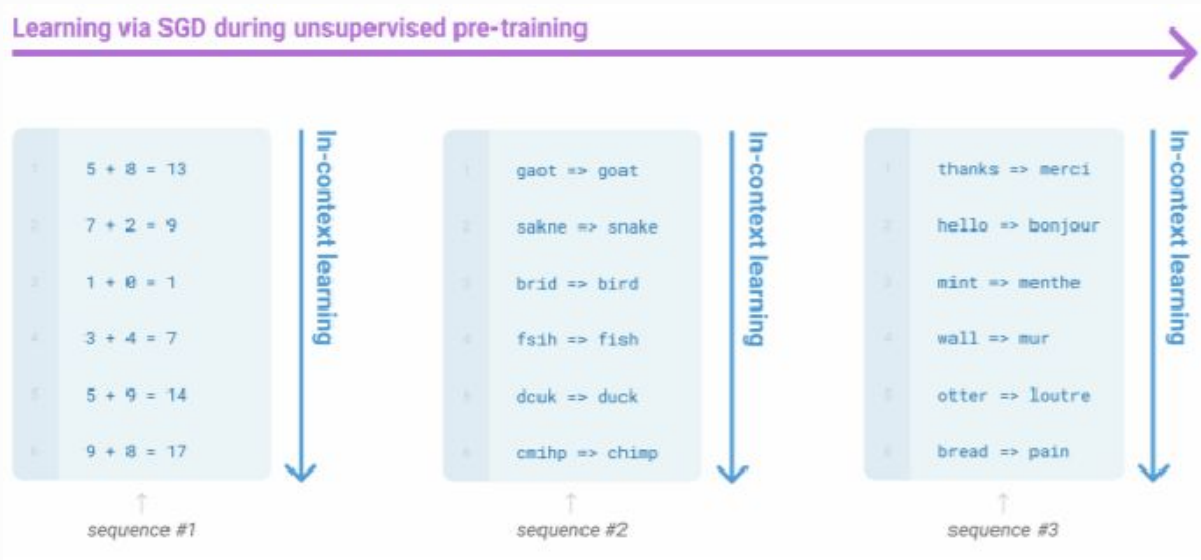
What kinds of things does pretraining teach?

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 Fibonnaci sequence]

· Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

LLMs as cultural technologies [Yiu et al. 2023]

- Example of innovating with tools biased against ChatGPT since little concept of physical world, can innovate and apply language to new settings (Tom)
    - But explores to find the causal relationships of the world? No
    - Imitation can provide incredible abilities (Max, Marcelo)
    - "You need intelligence even for copying" (Bhiman)
- Why innovate if you don't have desires and needs like humans? (Yixiao)
- Yes, they are cultural technologies to pass on information, but also are designed to be mimic intelligence. "Can intelligence be communicated culturally?" (RJ)
    - Novelty comes from the world, not agents (Ben)
    - Role in affecting us: how we interact, how we share information (Norah)
- Current shortcomings of LLMs
    - They don't automatically re-trained for data drift (Bhiman)
    - Check their own facts (Ben)
    - Learn rules from fewer examples, with the example of bias (Lokesh)
    - How do we improve AI models without so much data/capacity? (Gina)
- LLMs aren't using language to accomplish a task (like making a good love letter to elicit feelings) but just to match training set (Birju)

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]
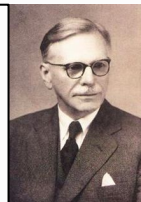
> "the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously"
> [Firth 1935]
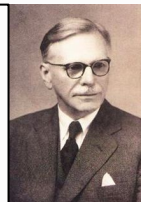
Let's use LLMs like BERT to get contextual word and sentence embeddings!

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
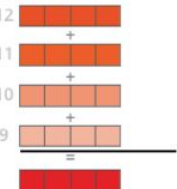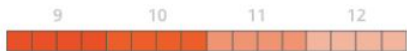
Slide adapted from David Mortensen
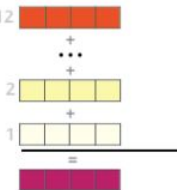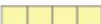
How Do You Get Embeddings from BERT?

- BERT_{BASE} 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
First Layer	91.0
Last Hidden Layer	94.9
Sum All 12 Layers	95.5
Second-to-Last Hidden Layer	95.6
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1

Wrapping up

- Pretraining NLP models to reconstruct text input on lots of data and then finetuning them on specific tasks is effective

- Pretrained models have different architectures for different types of tasks:

  - BERT: encoders of input (bidirectional, allows future context)

  - T5: encoder-decoder

  - GPT: decoder (causal/language modeling, can't see future context)

- You can use BERT to get nice contextual embeddings for words or sentences

Questions?