香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

DDA4220 Deep Learning and Applications

Lecture 9 Pretrained Language Model

Ruimao Zhang
zhangruimao@cuhk.edu.cn

School of Data Science
The Chinese University of Hong Kong (Shenzhen)

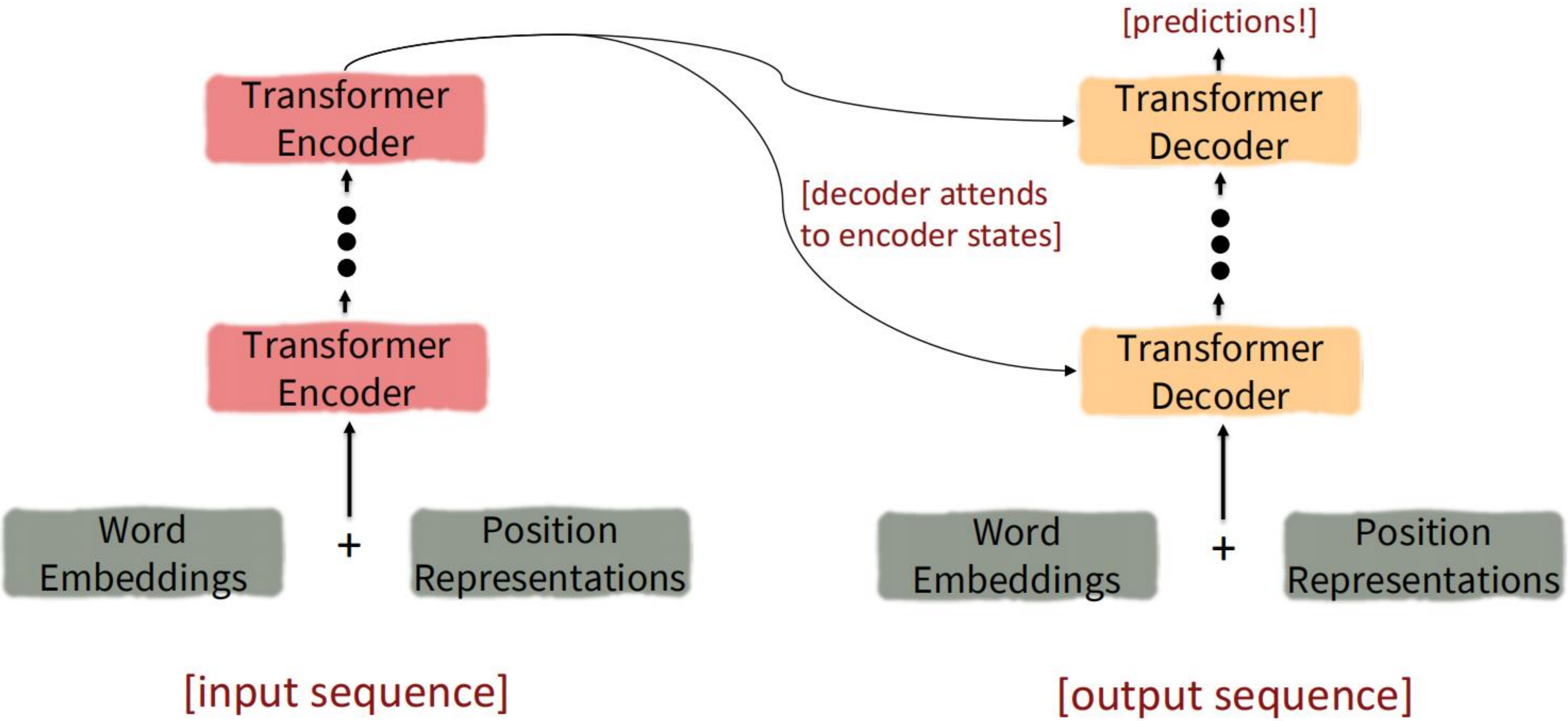
Slides partially credited to CS224n: Natural Language Processing with Deep Learning at Stanford / Winter 2022

Outline

- Quick review of Transformer model
- Motivating model pretraining from word embeddings
- Model pretraining three ways
  - Decoder-based
  - Encoder-based
  - Encoder-Decoder-based
- Very large models and in-context learning

The Transformer Encoder-Decoder

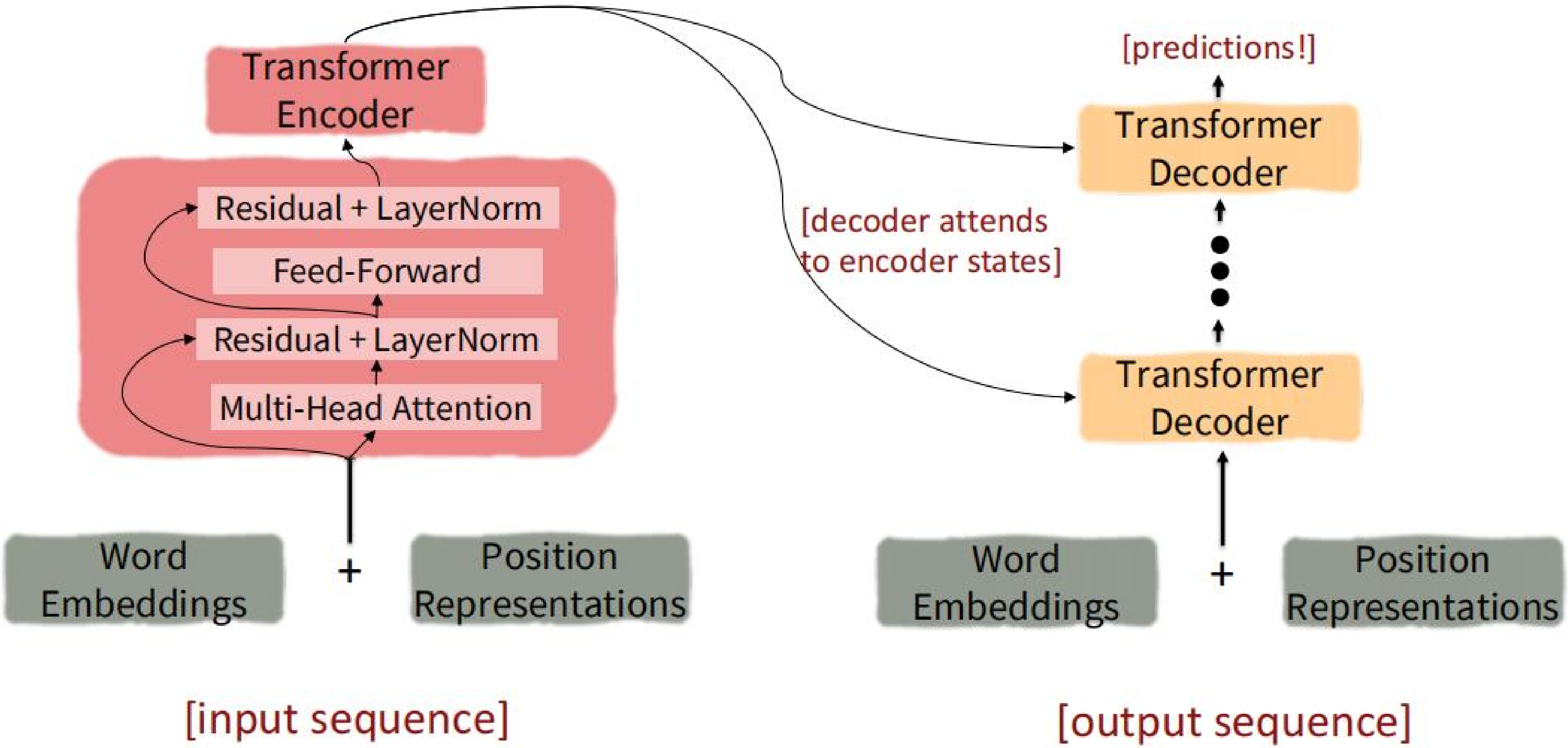
The basic architecture of the Transformer:



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

The Transformer Encoder-Decoder

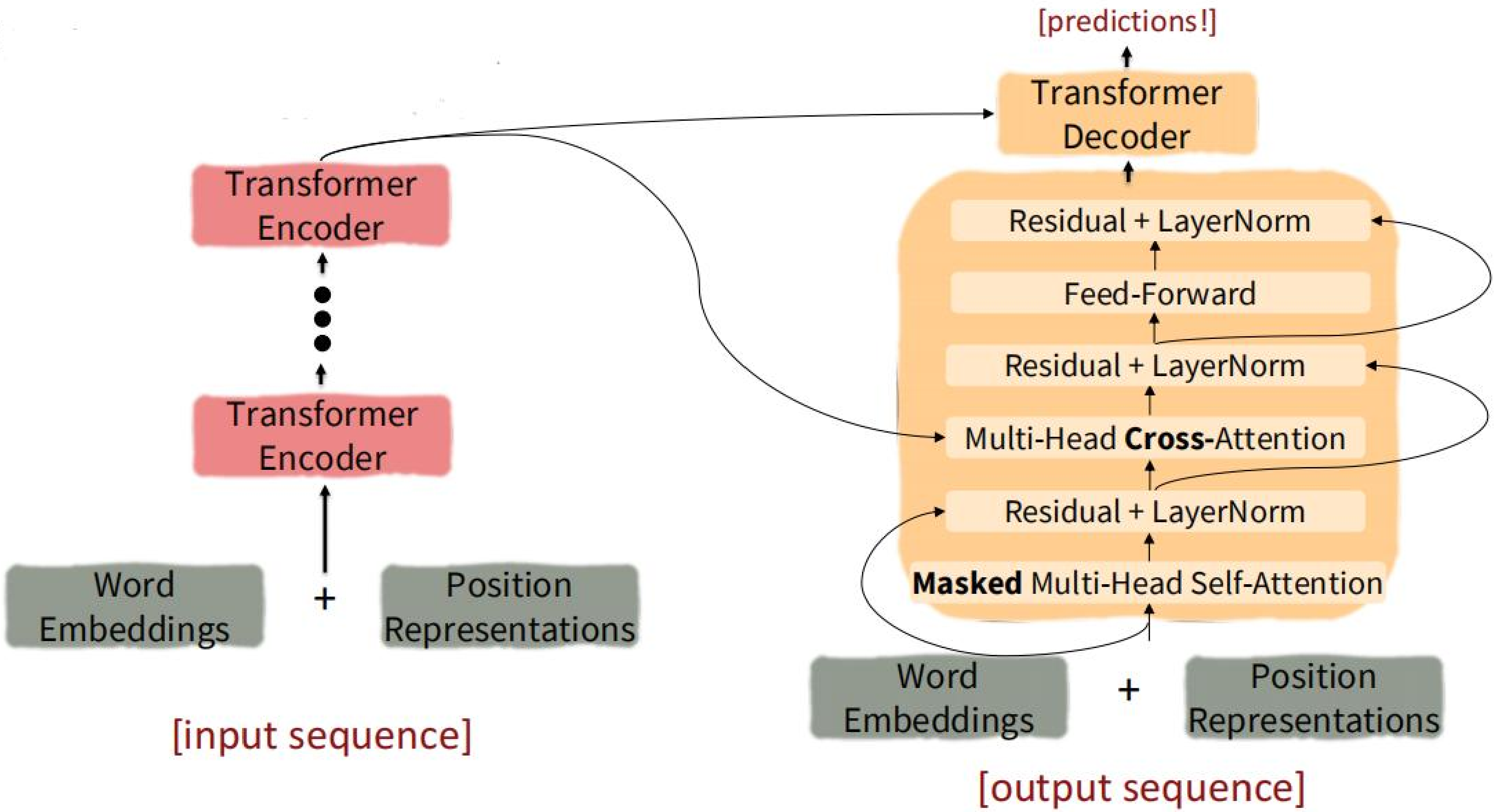
Looking back at the whole model, zooming in on an Encoder block:



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

The Transformer Encoder-Decoder

Looking back at the whole model, zooming in on a Decoder block:



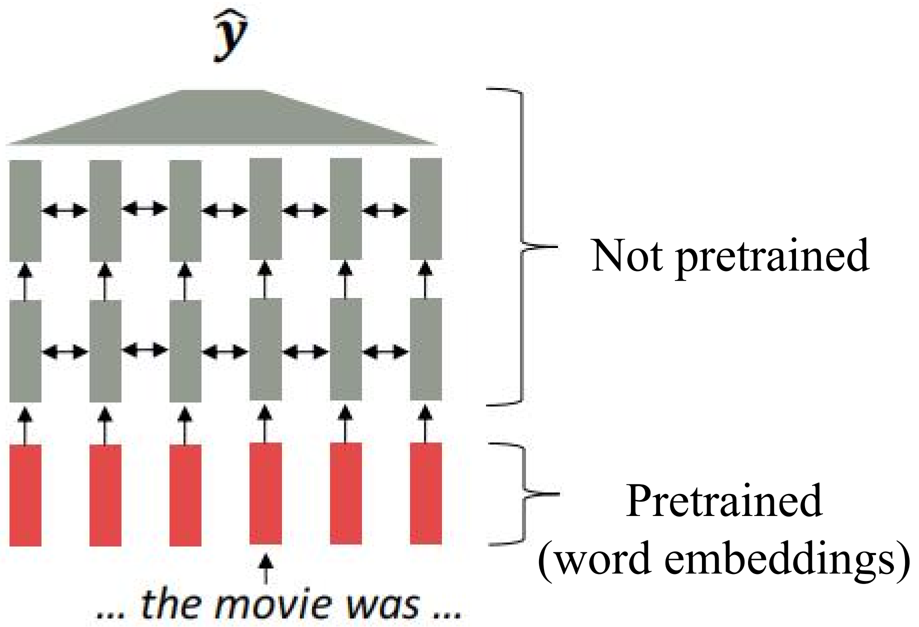
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

Where we were: pretrained word embeddings

- 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:

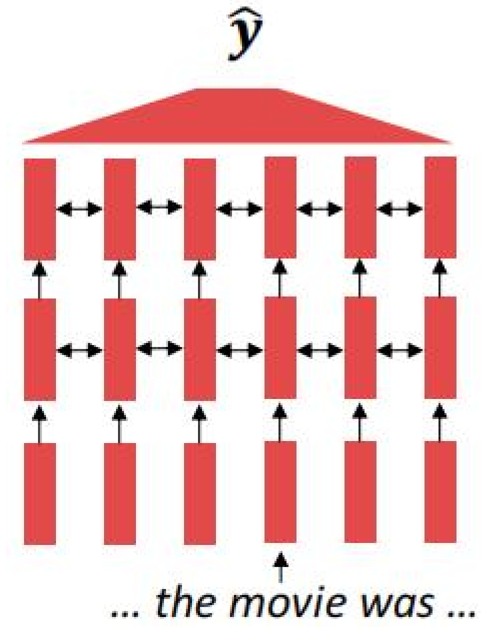
- The training data we have for our downstream task (like question answering) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network are randomly initialized!



Where we're going: pretraining whole models

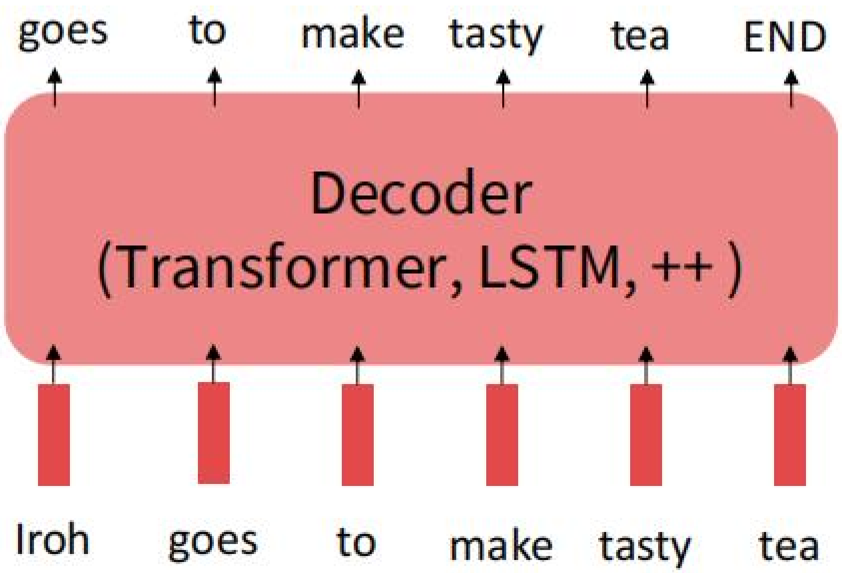
In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining.**
- Pretraining methods hide parts of the input from the model, and then 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



The model has learned how to represent entire sentences through pretraining

Pretraining through language modeling



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!

Pretraining through language modeling:

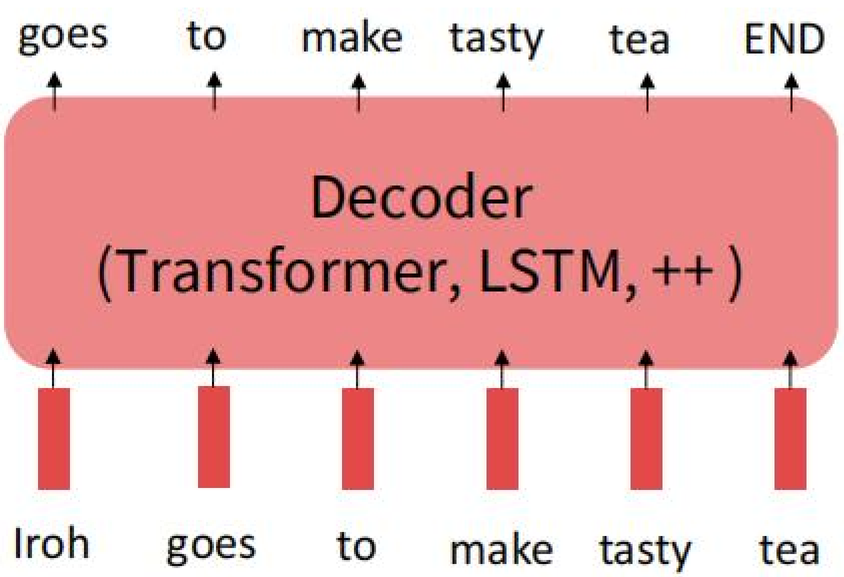
- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.

The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter

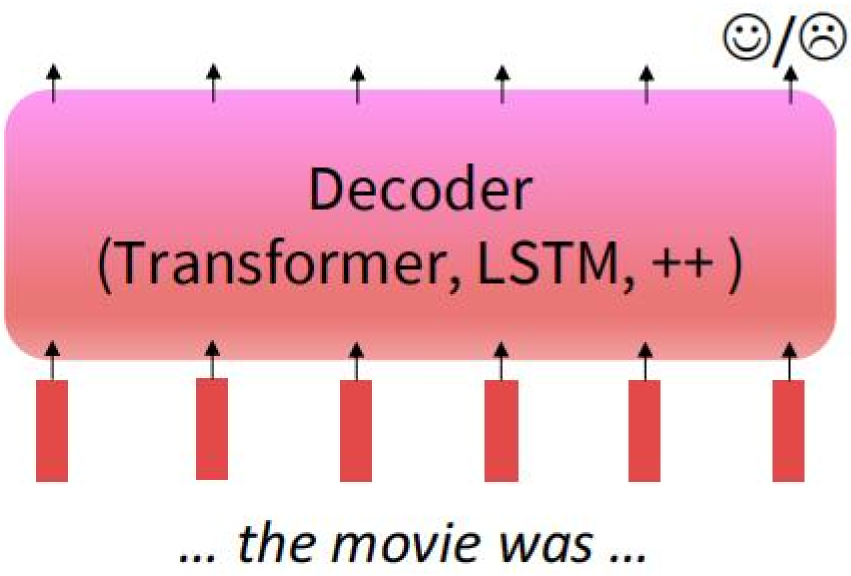
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



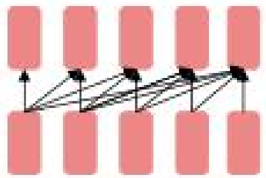
Step 2: Finetune (on your task)

Not many labels; adapt to the task!



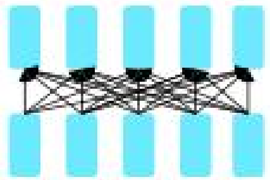
Pretraining for three types of architectures

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



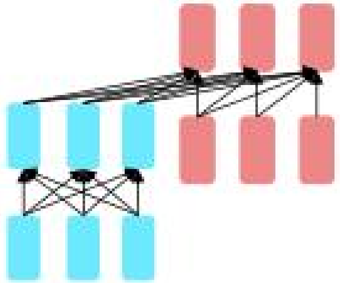
Decoders

- Language models! What we‘ve seen so far.
- Nice to generate from; can’t condition on future words
- **Examples:** GPT, GPT-2, GPT-3



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- **Examples:** BERT and its many variants



Encoder-Decoders

- Good parts of decoders and encoders?
- What‘s the best way to pretrain them?
- **Examples:** Transformer, T5

Transformer, BERT and GPT

OpenAI

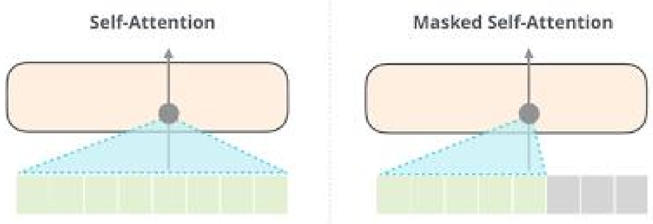
Transformer 06/2017 — Attention Is All You Need

GPT 06/2018 — Improving Language Understanding by Generative Pre-Training

BERT 10/2018 — BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

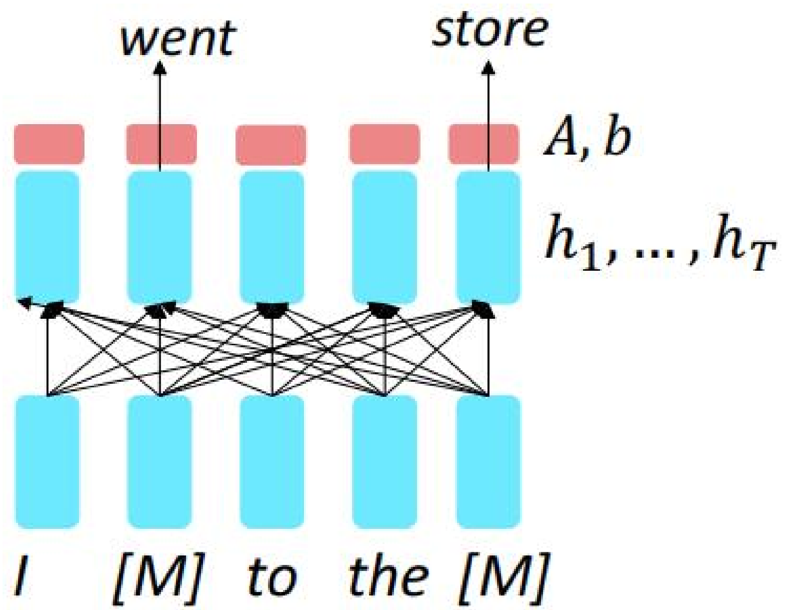
GPT-2 02/2019 — Language Models are Unsupervised Multitask Learners

GPT-3 05/2020 — Language Models are Few-Shot Learners



Pretraining encoders: what pretraining objective to use?

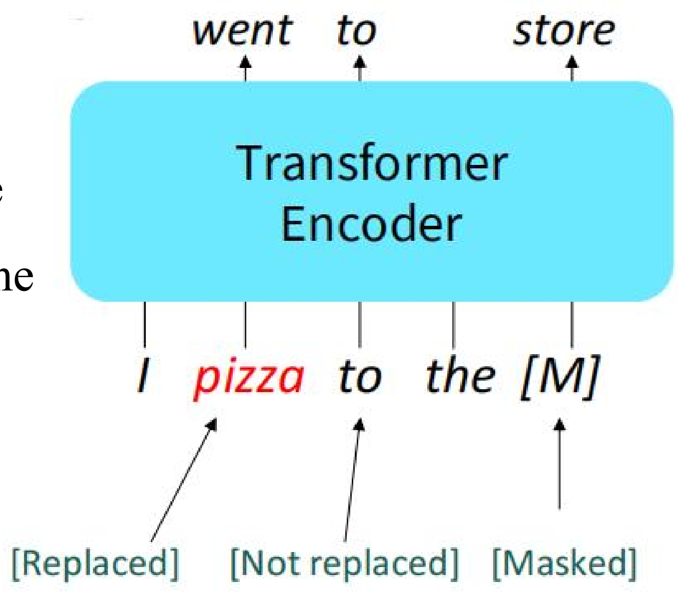
- So far, we've looked at language model. **But encoders get bidirectional context**, so we can't do language modeling!
- **Idea:** replace some fraction of words in the input with a special **[MASK]** token; predict these words.
- Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$



BERT: Bidirectional Encoder Representations

- Devlin et al., 2018 proposed the "Masked LM" objective, and **released the weights of their pretrained Transformer (BERT)**.

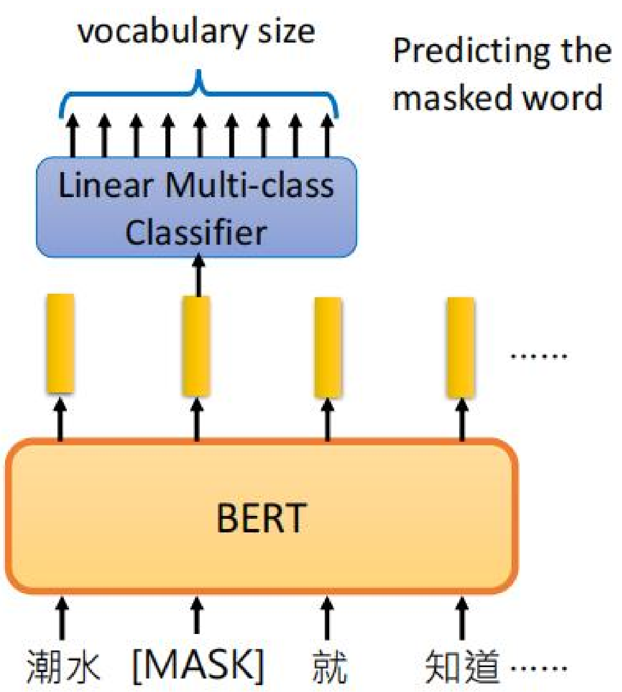
- Some more details about Masked LM for BERT:
- Predict a random 15% of (sub)word tokens.
    - Replace input word with [MASK] 80% of the time
    - Replace input word with a random token 10% of the time
    - Leave input word unchanged 10% of the time (but
    - still predict it!)
- Why? Doesn't let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



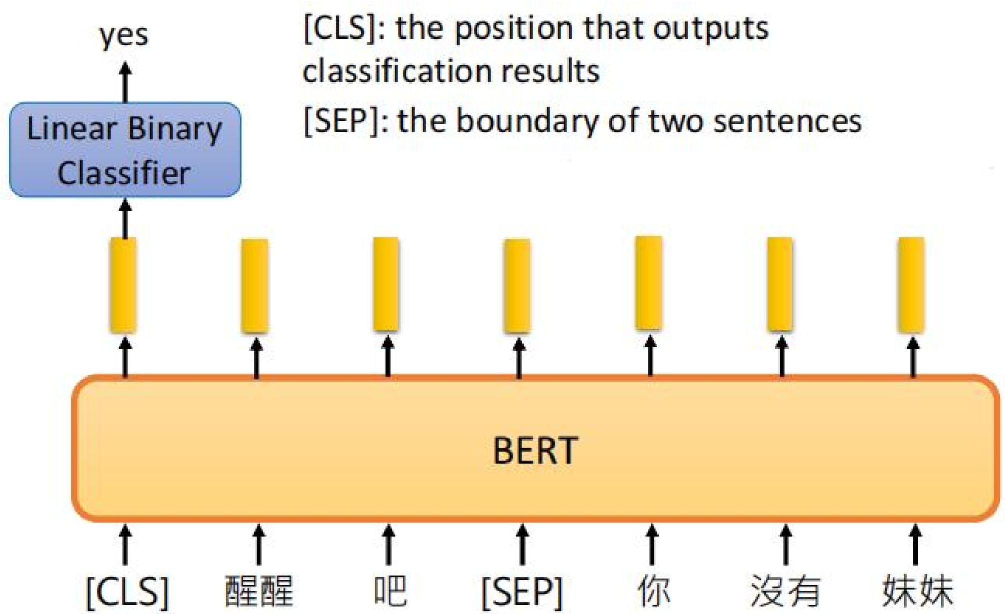
Training of BERT

- Approaches 1 and 2 are used at the same time to train the BERT.

Approach 1: Masked LM

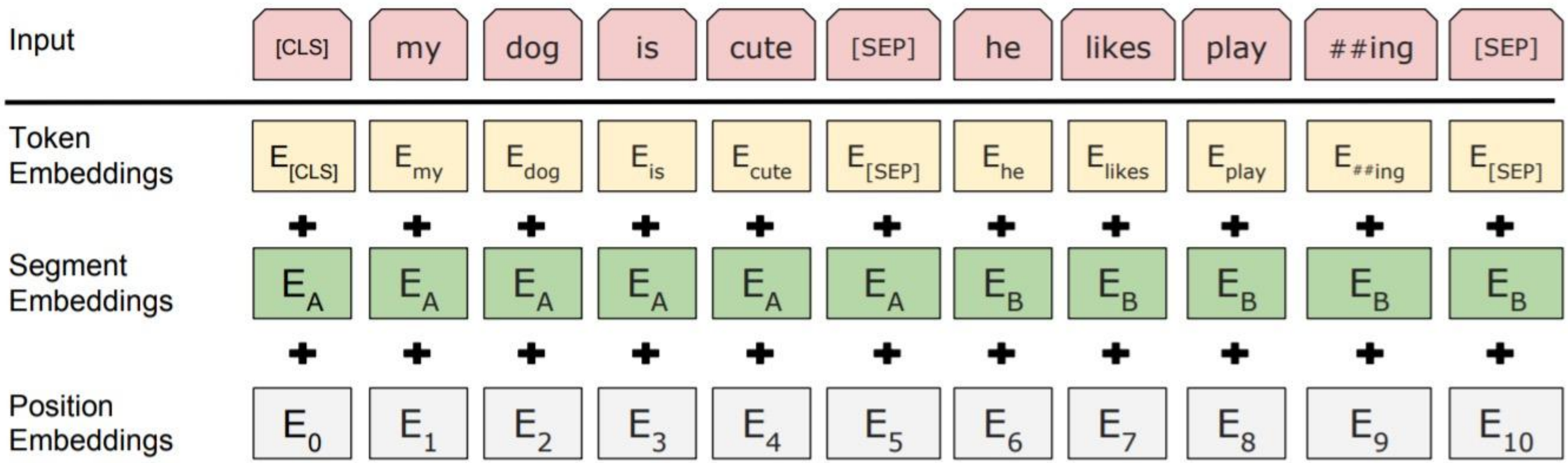


Approach 2: Next Sentence Prediction



BERT: Bidirectional Encoder Representations

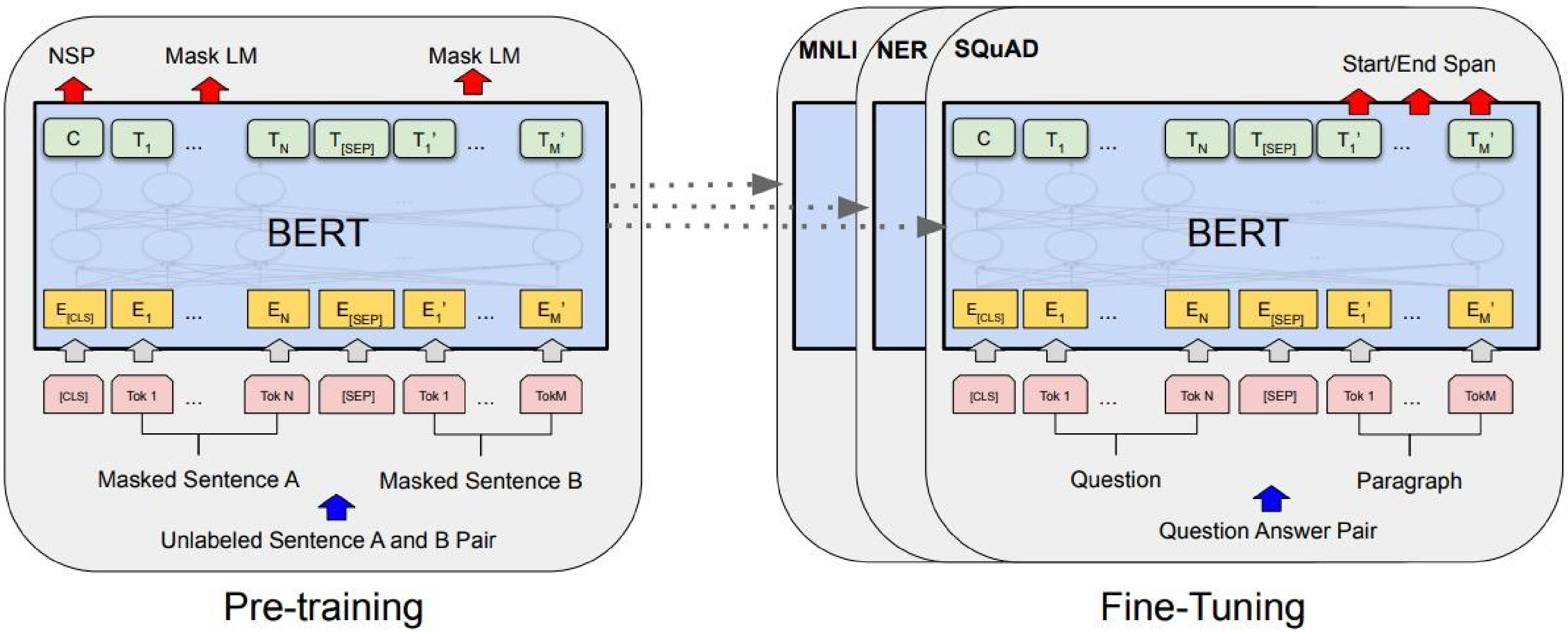
- The pretraining input to BERT was two separate contiguous chunks of text:



- BERT was trained to predict whether one chunk follows the other or is randomly sampled.

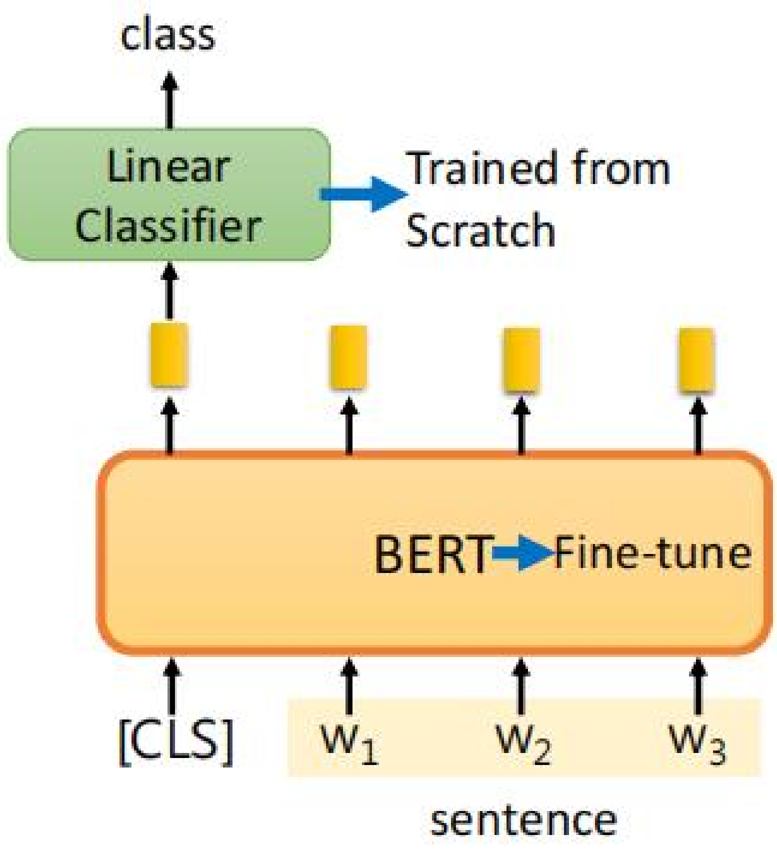
BERT: Bidirectional Encoder Representations

- **Unified Architecture:** As shown below, there are minimal differences between the pre-training architecture and the fine-tuned version for each downstream task.



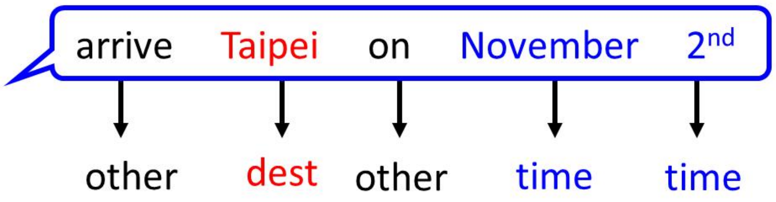
How to use BERT – Case 1

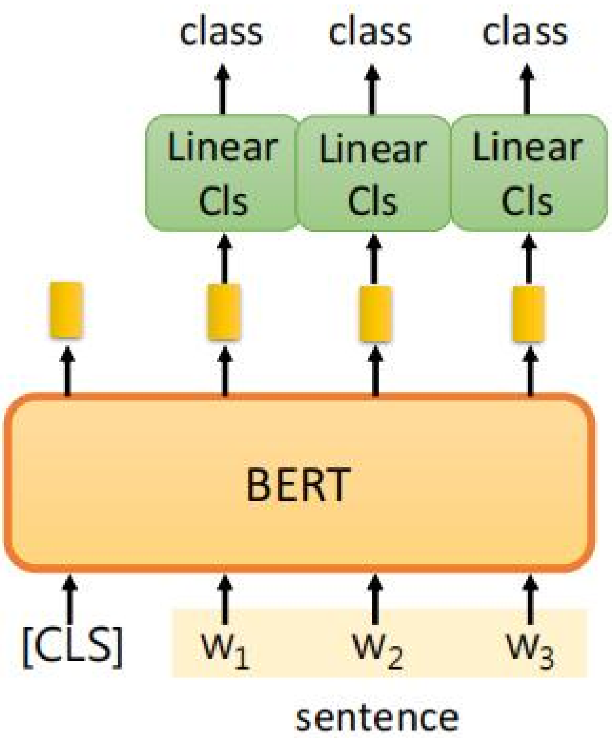
- Input: single sentence, output: class
- Example: Sentiment analysis (our HW), Document Classification



How to use BERT – Case 2

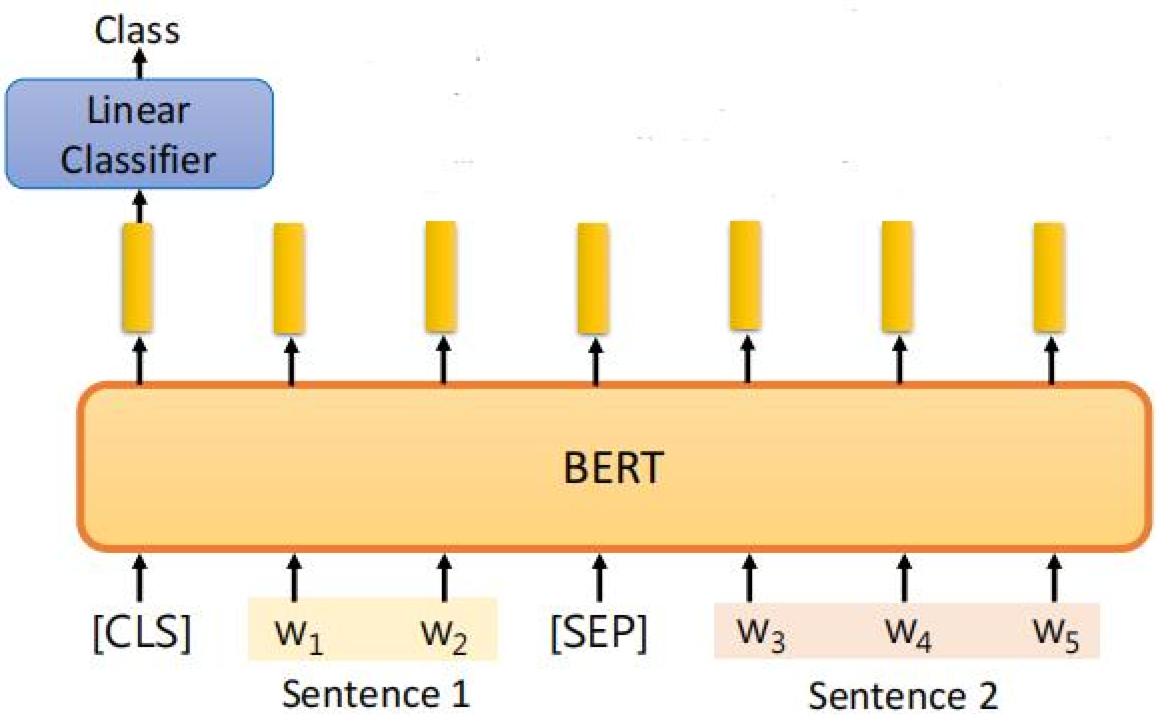
- Input: single sentence, output: class of each word
- Example: Slot filling





How to use BERT – Case 3

- Input: two sentences, output: class
- Example: Natural Language Inference
  - Given a "premise", determining whether a "hypothesis" is T/F/ unknown.



The SOTA performance of BERT

- BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

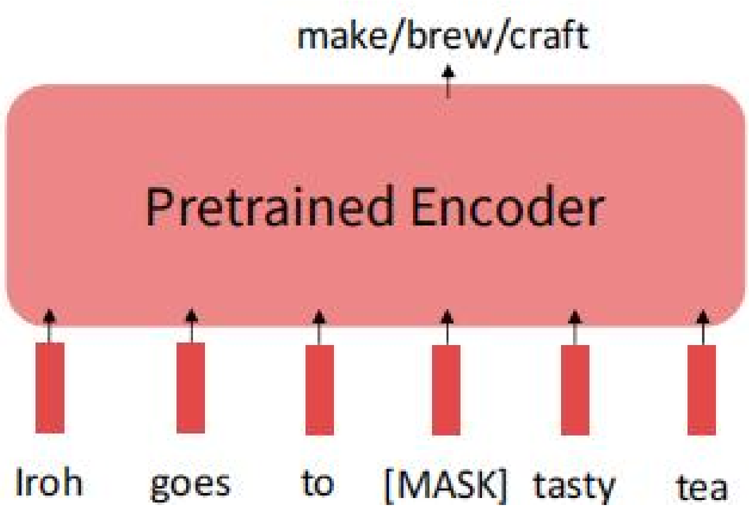
- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI:** natural language inference over question answering data
- **SST-2:** sentiment analysis
- **CoLA:** corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B:** semantic textual similarity
- **MRPC:** microsoft paraphrase corpus
- **RTE:** a small natural language inference corpus

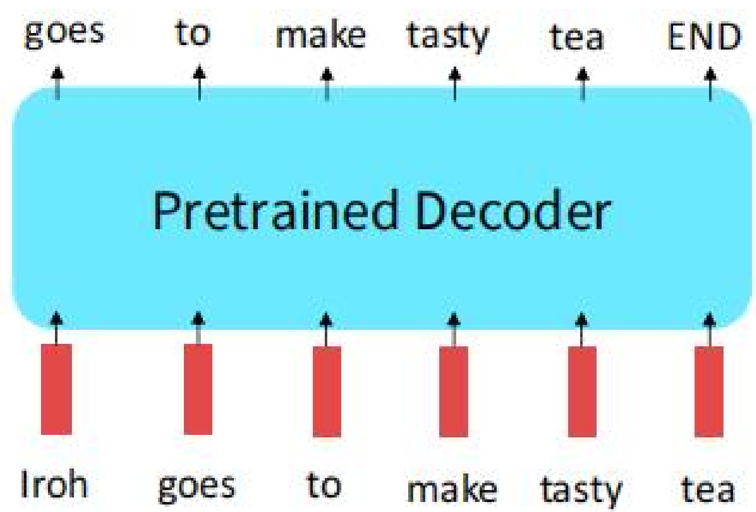
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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**

Note that BERTBASE was chosen to have the same number of parameters as OpenAI GPT.

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.



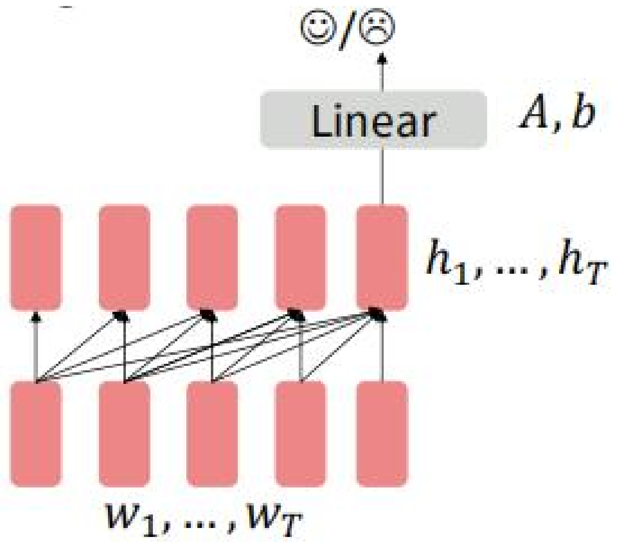


Pretraining decoders

- 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.

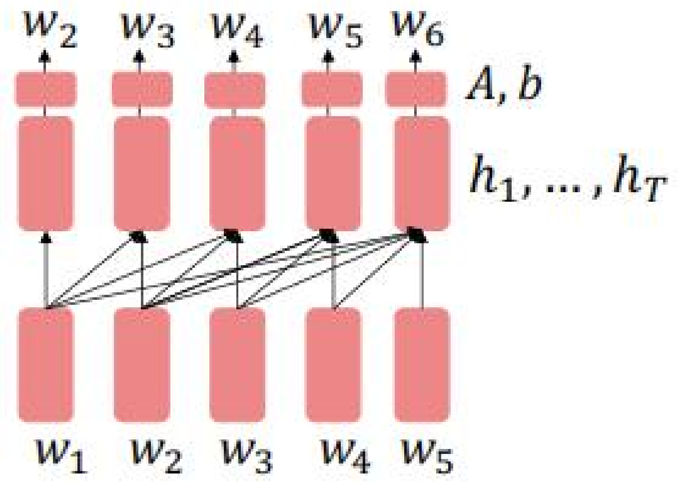
Note the linear layer hasn't been pretrained and must be learned from scratch.

Pretraining decoders

- It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$
- This is helpful in tasks where the output is a sequence with a vocabulary like that at pretraining time!
  - Dialogue (context=dialogue history)
  - Summarization (context=document)

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

Where A and b were pretrained in the language model!

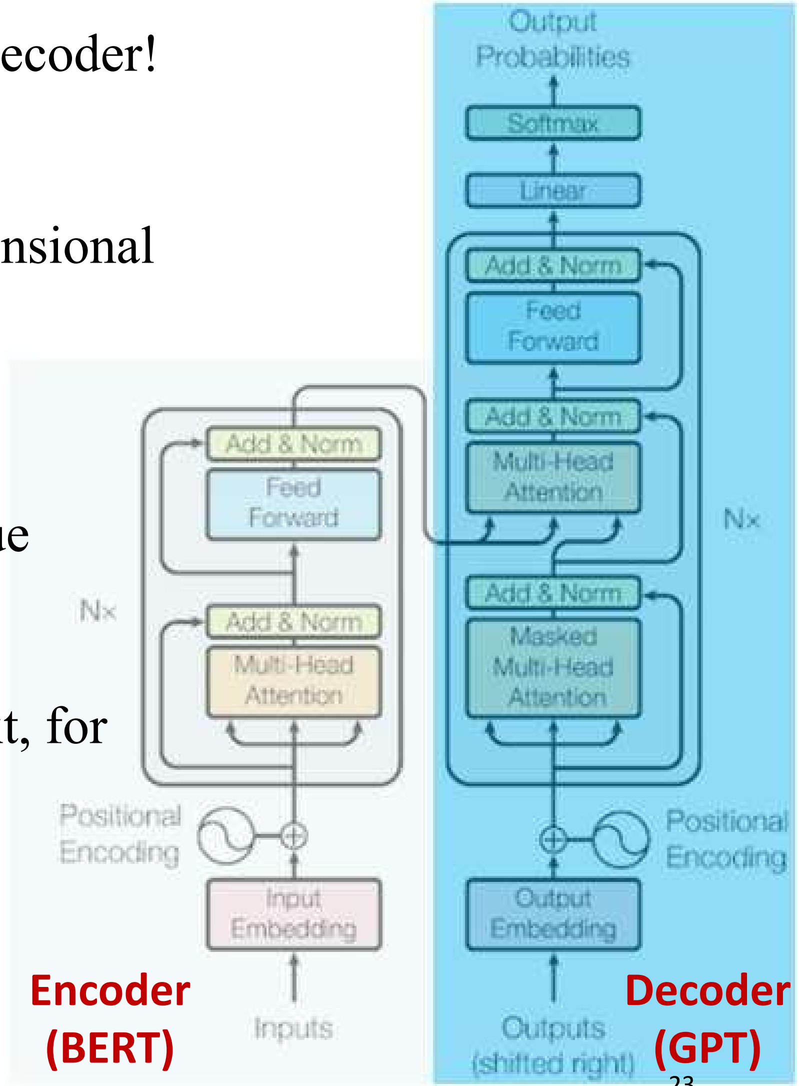


Note thethe linear layer has been pretrained.

Generative Pretrained Transformer (GPT)

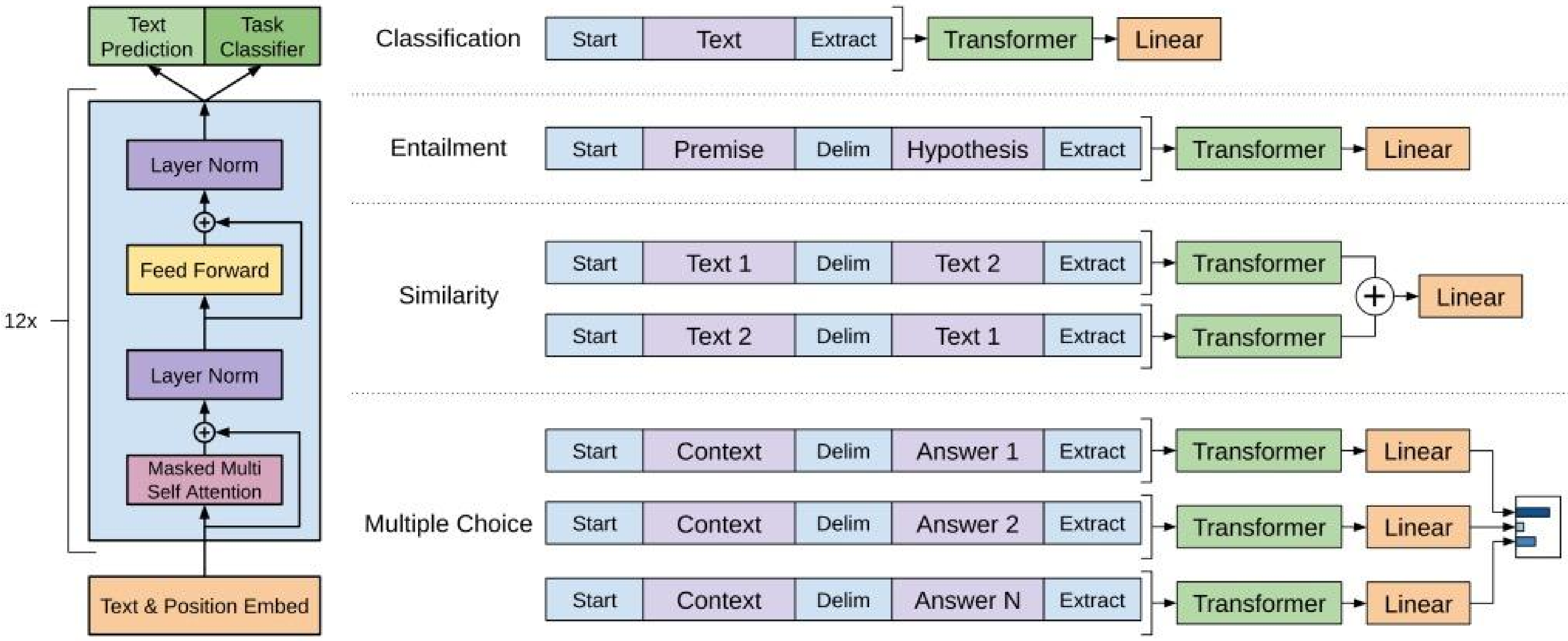
2018's GPT was a big success in pretraining a decoder!

- Transformer decoder with 12 layers
- 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.



Generative Pretrained Transformer (GPT)

How do we format inputs to our decoder for finetuning tasks?



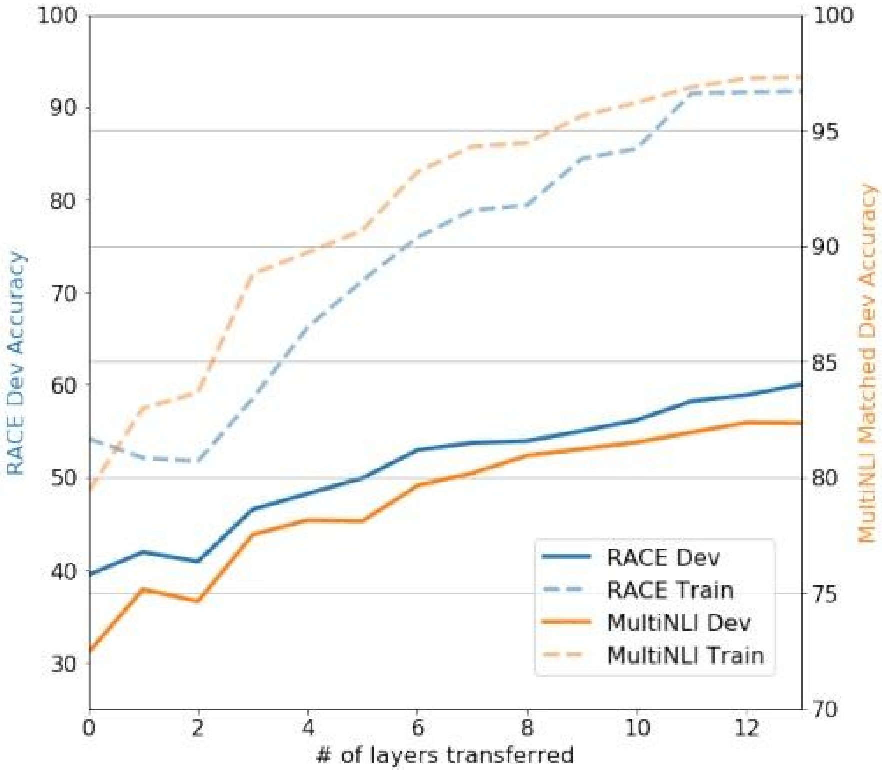
The linear classifier is applied to the representation of the [EXTRACT] token.

Generative Pretrained Transformer (GPT)

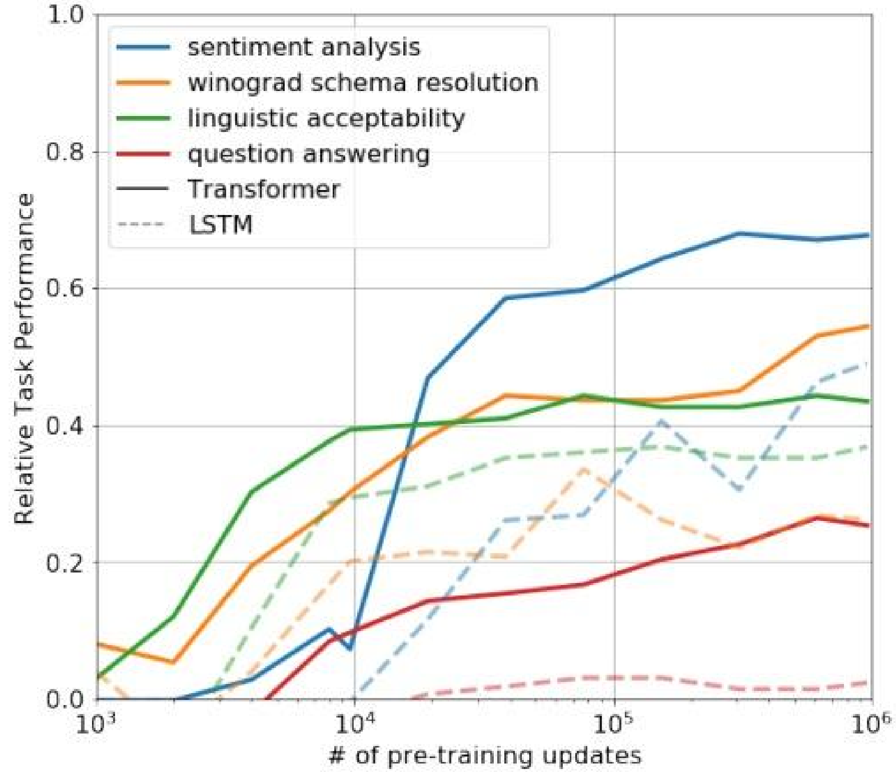
GPT results on various natural language inference datasets.

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

Examining the Effect of Pretraining in GPT



As more layers are transferred, performance improves on RACE (a large-scale reading comprehension dataset) and MultiNLI.



Zero-shot performance of Transformer vs. LSTM as a function of the # of pre-training updates.

Increasingly convincing generations (GPT2)

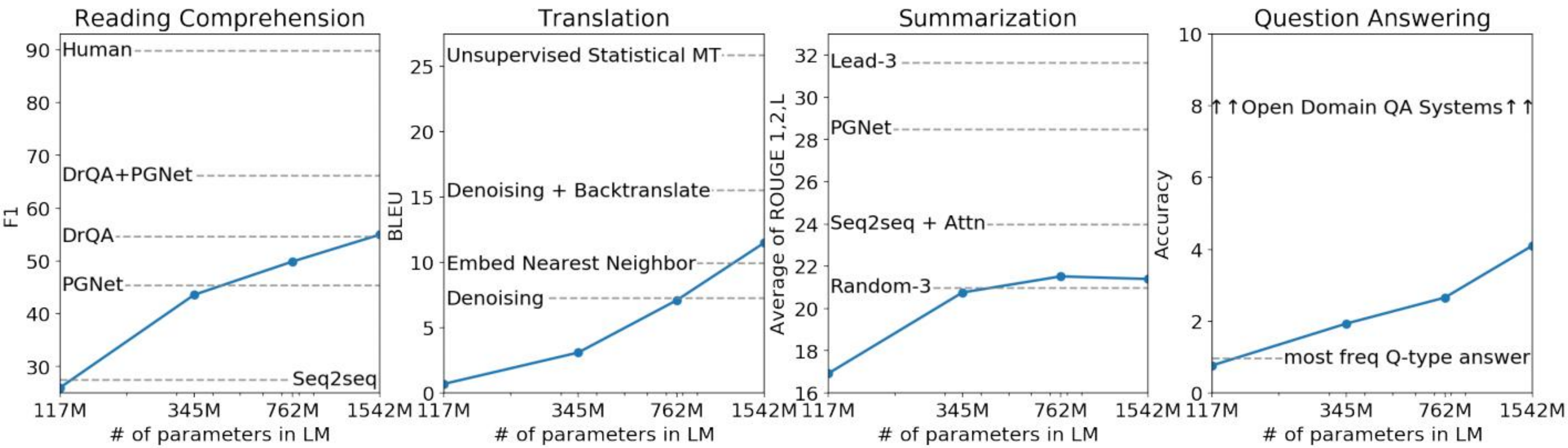
We mentioned how pretrained decoders can be used in their capacities as language models. GPT-2, a larger version of GPT trained on more data, was shown to produce relatively convincing samples of natural language

Model	Released Time	Parameters	Data
GPT	2018/06	0.17 B	about 5 GB
GPT-2	2019/02	1.5 B	40 GB
GPT-3	2020/05	175 B	45 TB

Architecture hyperparameters for the 4 model sizes.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Increasingly convincing generations (GPT2)

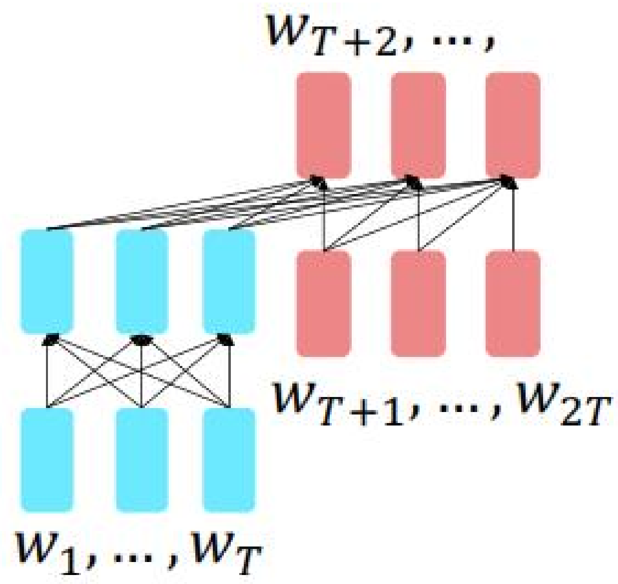


Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al.,2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019).

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.

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



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

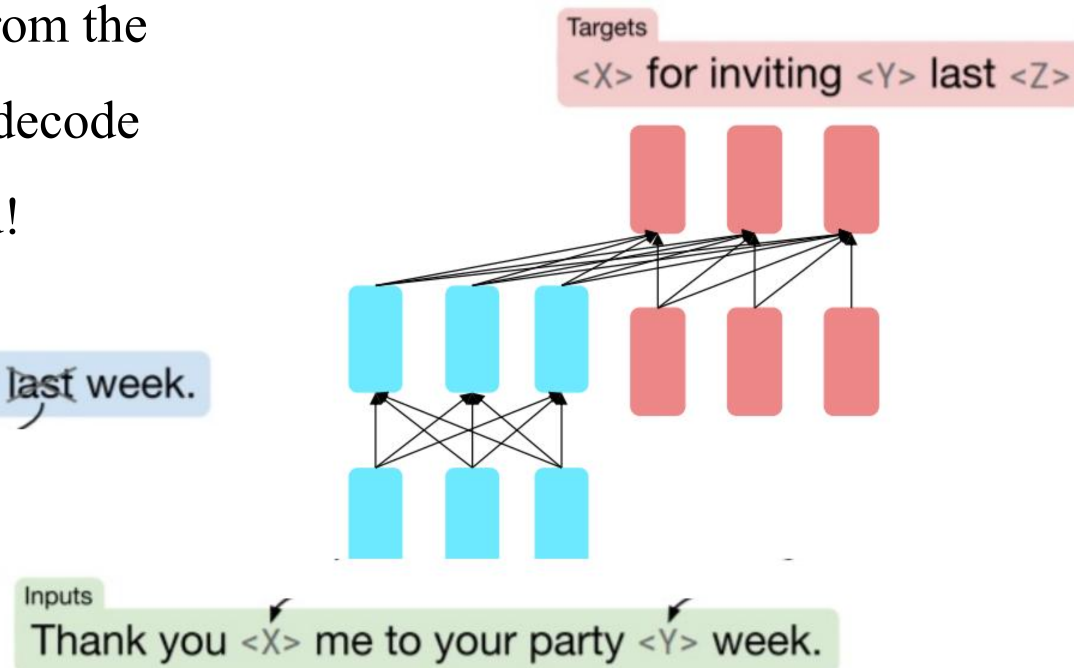
Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer." The Journal of Machine Learning Research 21, no. 1 (2020): 5485-5551.

Pretraining encoder-decoders

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!



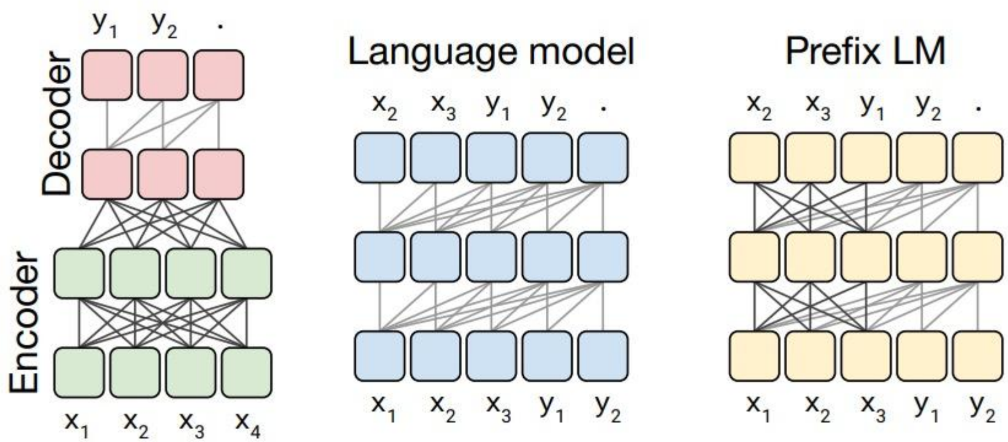
Original text

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.

Pretraining encoder-decoders

[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

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 then take their predictions.

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

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has 175 billion parameters.

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

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

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

Input (prefix within a single Transformer decoder context):

" thanks -> merci
hello -> bonjour
mint -> menthe
otter -> "

Output (conditional generations):

loutre..."

Task-agnostic Language Model

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:      ← task description
cheese => ........                ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:      ← task description
sea otter => loutre de mer        ← example
cheese => ........                ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:      ← task description
sea otter => loutre de mer        ← examples
peppermint => menthe poivrée      ←
plush girafe => girafe peluche    ←
cheese => ........                ← prompt
```

Traditional fine-tuning (not used for GPT-3)

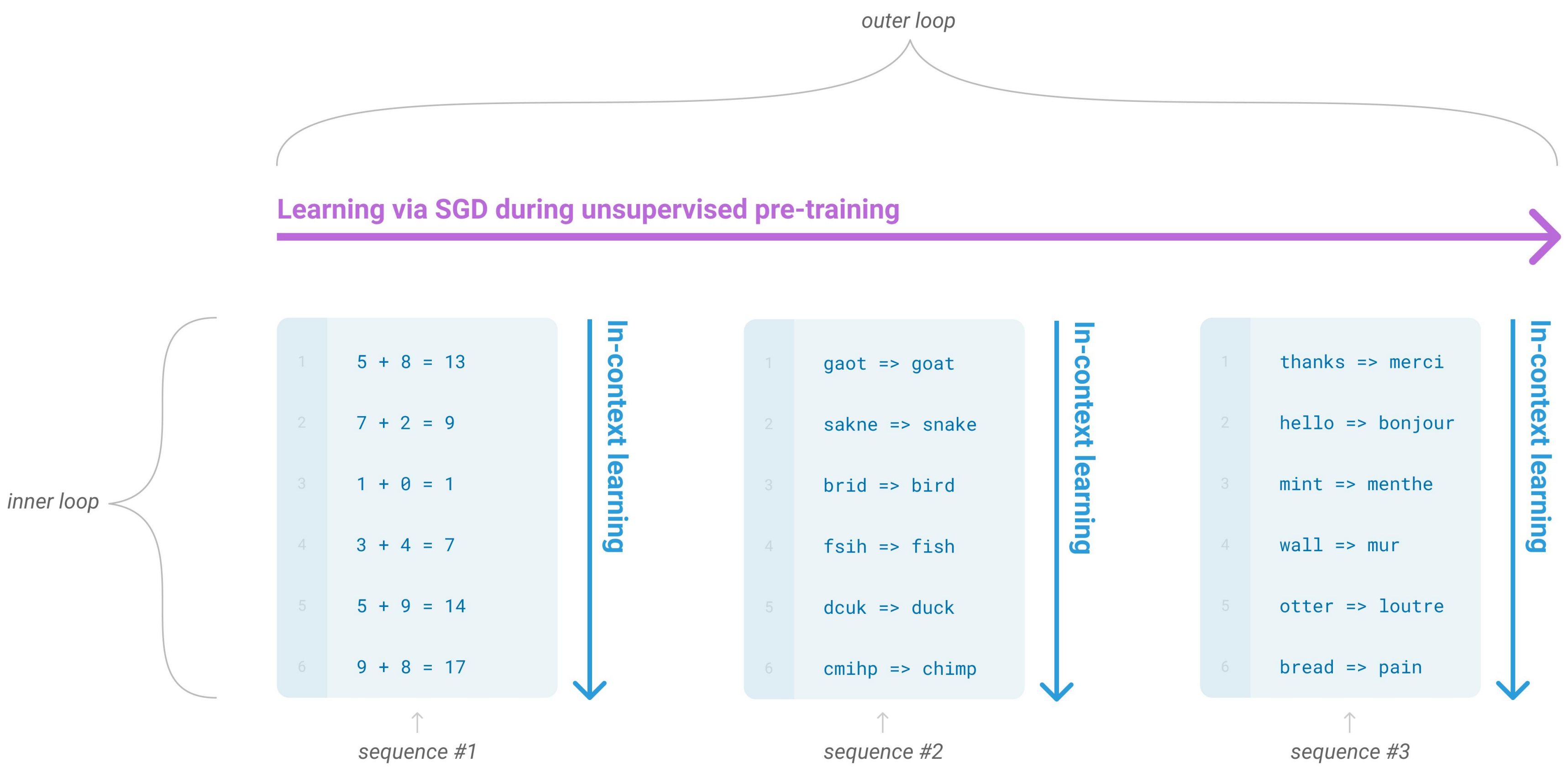
Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

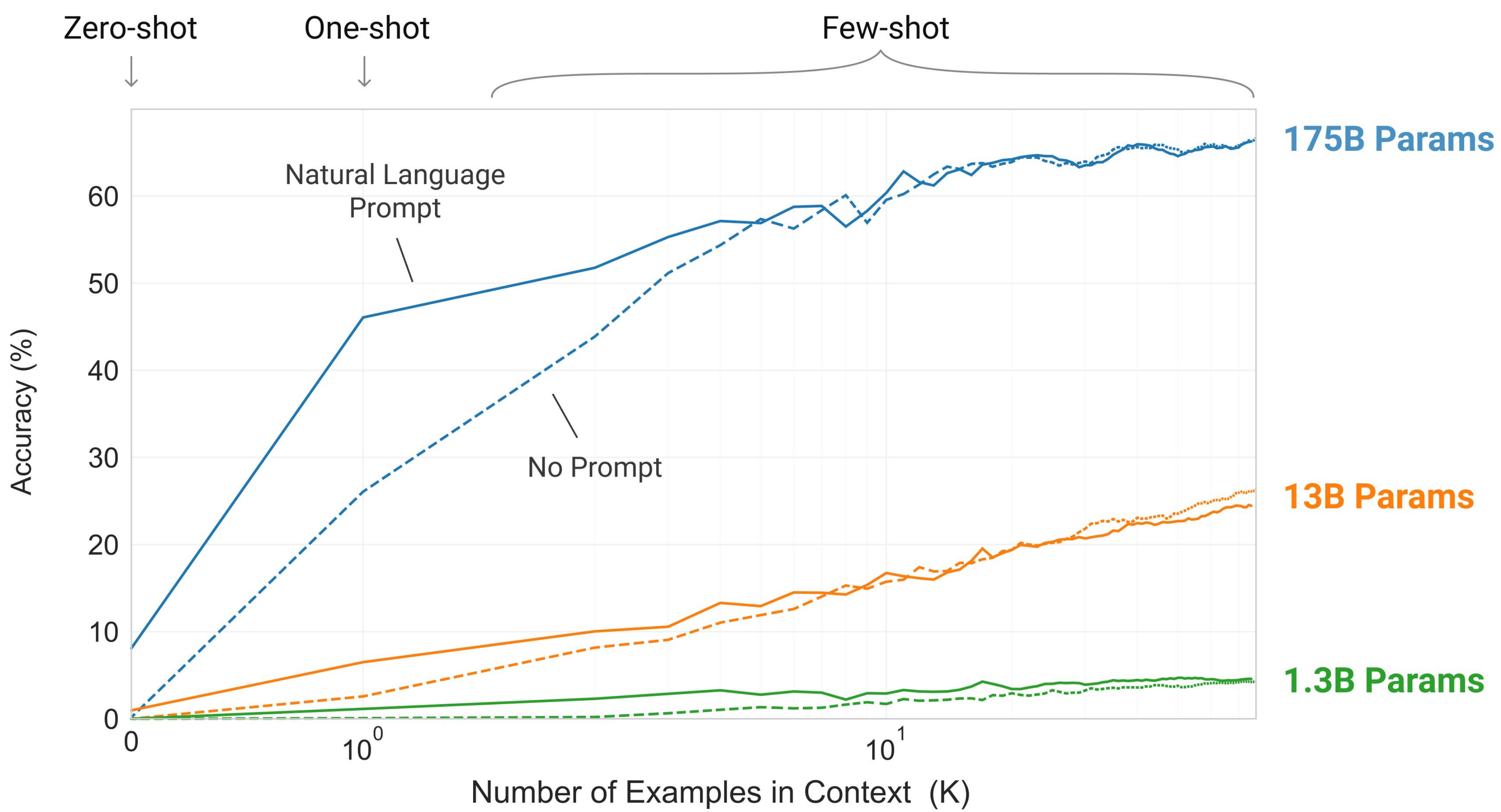
```
sea otter => loutre de mer        ← example #1
        ↓
  gradient update
        ↓
peppermint => menthe poivrée      ← example #2
        ↓
  gradient update
        ↓
       ...
        ↓
plush giraffe => girafe peluche   ← example #N

  gradient update

cheese => ........                ← prompt
```

In-context Learning

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



Larger models make increasingly efficient use of in-context information.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Thanks for Listening !

Ruimao Zhang

Room 517, Daoyuan Building, The Chinese Univeristy of Hong Kong, Shenzhen
zhangruimao@cuhk.edu.cn
ruimao.zhang@ieee.org