PAUL G. ALLEN SCHOOL

UNIVERSITY of WASHINGTON

NLP and LMs

Pre-training

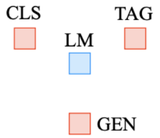
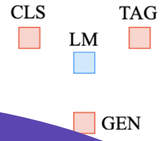
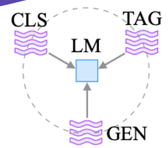
Luke Zettlemoyer

Slides adapted from Kabir Ahuja, Yejin Choi,

Liwei Jiang, John Hewitt, Anna Goldie

Major Paradigms in NLP

Liu et al. 2021

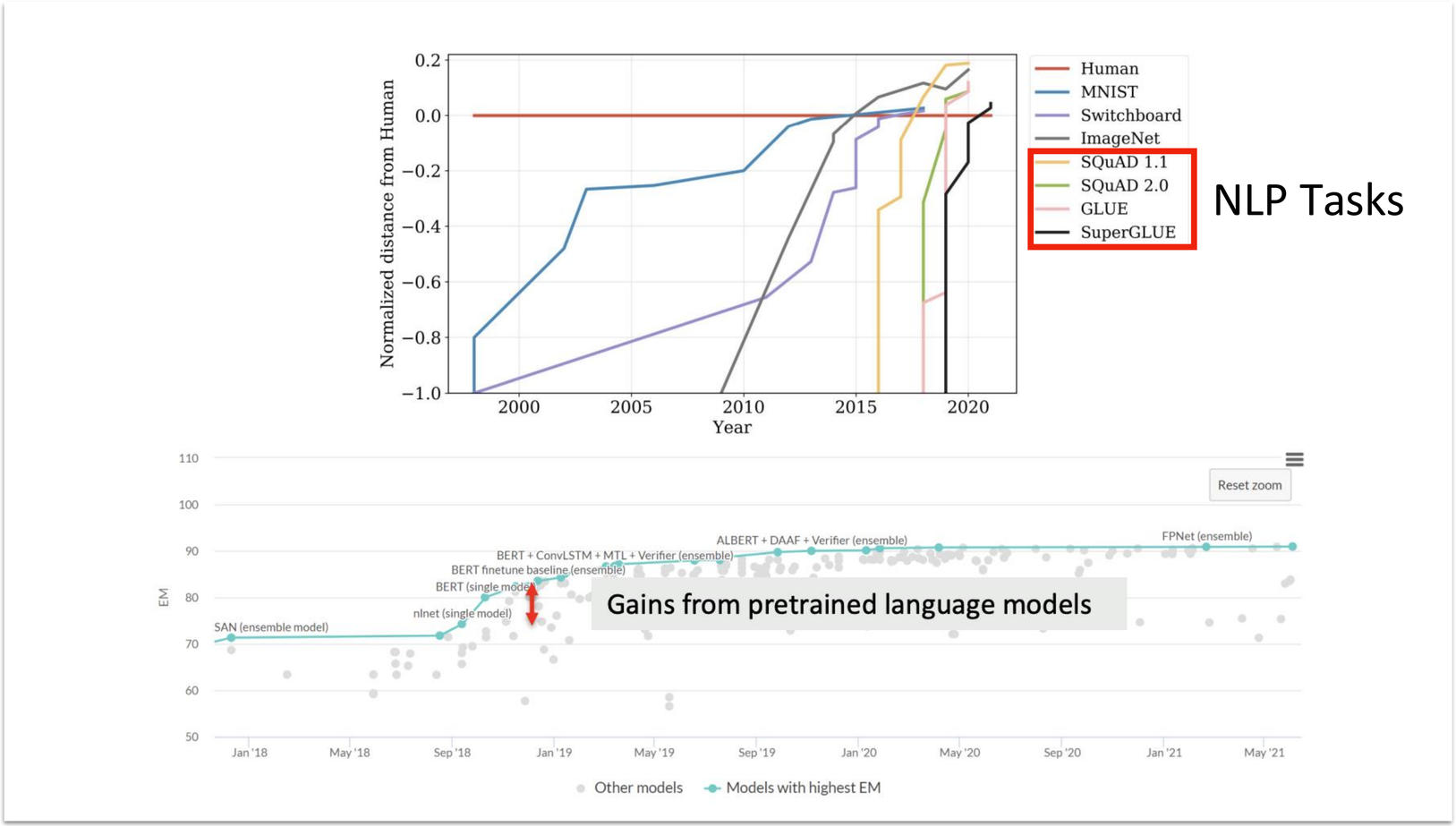
Paradigm	Engineering	Task Relation	
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)		Pre 2017
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)		
c. Pre-train, Fine-tune			2017-2019
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)		2021- Present?
e. Pre-train, Alignment (RLHF), (Fine-tune), Predict			2022- Present
f. Pre-train, Alignment (RLHF), (Fine-tune), RL for Reasoning, Predict			2022- Present

What we have seen so far

Pre-training common across all major paradigms post 2017

What we will see in the coming lectures

The Pre-training Revolution



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

Slide from Chris Manning, Lecture 9: Pre-training, CS224n Spring 2024

Lecture Outline

1. Motivating Pre-training, aka Self-supervised Learning
2. Pre-training Architectures and Training Objectives
    1. Encoders
    2. Encoder-Decoders
    3. Decoder

Lecture Outline

1. Motivating Pre-training, aka Self-supervised Learning
2. Pre-training Architectures and Training Objectives
    1. Encoders
    2. Encoder-Decoders
    3. Decoder

Issues with Fully Supervised Learning Approaches

Food Review: “I recently had the pleasure of dining at Fusion Bites, and the experience was nothing short of spectacular. The menu boasts an exciting blend of global flavors, and each dish is a masterpiece in its own right.”

Say that we are given a dataset of 100K food reviews with sentiment labels, **how do we train a model to perform sentiment analysis over unseen food reviews?**

We can directly train a randomly initialized model to take in food review texts and output “positive” or “negative” sentiment labels.

Issues with Fully Supervised Learning Approaches

Food Review: “I recently had the pleasure of dining at Fusion Bites, and the experience was nothing short of spectacular. The menu boasts an exciting blend of global flavors, and each dish is a masterpiece in its own right.”

Movie Review: "The narrative unfolds with a steady pace, showcasing a blend of various elements. While the performances are com… …tography captures the essence of the story, the…

Fully Supervised Learning
Collect a labeled dataset for movie reviews and train a model from scratch on this new dataset

If we are instead given **movie revi…**
trained from food reviews to predict t…

May NOT generalize well due to distributional shift!

Transfer Learning: A History Lesson from Computer Vision

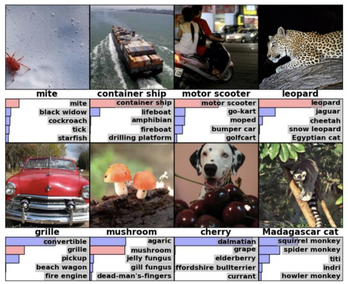
- Instead of training a randomly initialized neural network every time we encounter a new task or domain,
  - can we re-use the learned representations from one task/domain for another?

Idea: Train a **(very) deep neural network** on a **large-scale dataset** and re-use the learned representations from this network to adapt to new tasks

ImageNet Challenge

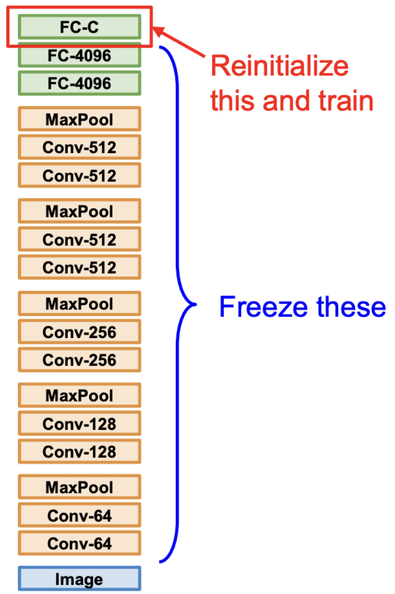
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

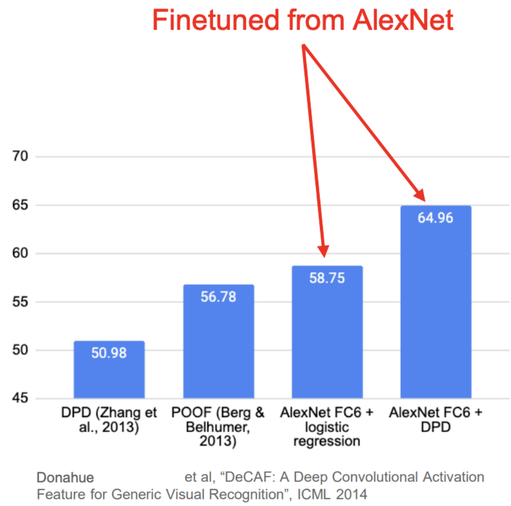


1. Train on Imagenet

2. Small Dataset (C classes)



This is called Fine-tuning!



Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

- A very successful recipe for adapting to different vision tasks like object detection, semantic segmentation, pose estimation, etc.
- Also, reduced the reliance on large training datasets to achieve good performance

Image from Lecture 7 CS231n slides by Fei-Fei Li, Ehsan Adeli, Zane Durante

Why it took so long for NLP?

- Since 2014, it had become common practice in the Computer Vision community to download a pre-trained (on Image Net) deep neural network model and “fine-tune” it on the problem at hand instead of starting from scratch.
- This wasn’t the case in NLP till late 2017s.
- It was common to use pre-trained word vectors like word2vec, GloVe for NLP tasks, and while those would help boost performance, most often it was a marginal improvement.

You might have seen this already while attempting HW2

NLP?

Part of input data itself provides labels instead of requring external labels. What SSL model have we already seen? Language Models!

What changed starting from 2017?

We can mostly boil down this delay t… …ctors

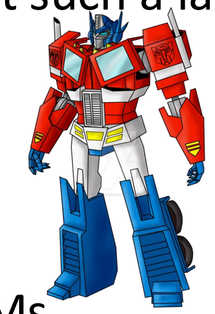
Self-supervised Learning

1. Lack of a large-scale general dataset

    1. It wasn't clear what would be a suitable NLP task most representative of the space of NLP tasks (classification, QA, NLI, Parsing, Language Modeling?). Getting high-quality label at such a large scale was also a challenge.

2. Neural Network Models for NLP were usually very shallow

Transformers



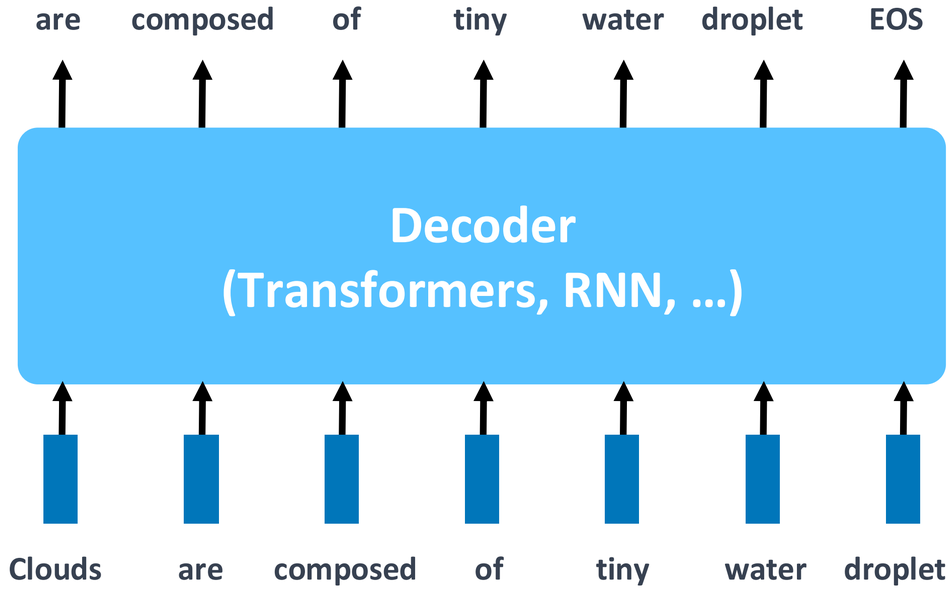
    1. Pre-2017, dominant models used in NLP were recurrent neural networks e.g. LSTMs

    2. These models were usually 1-2 hidden layers, and scaling them to a large number of layers was non-trivial as these models were notoriously hard to train

Self-supervised Pre-training for Learning Underlying Patterns, Structures, and Semantic Knowledge

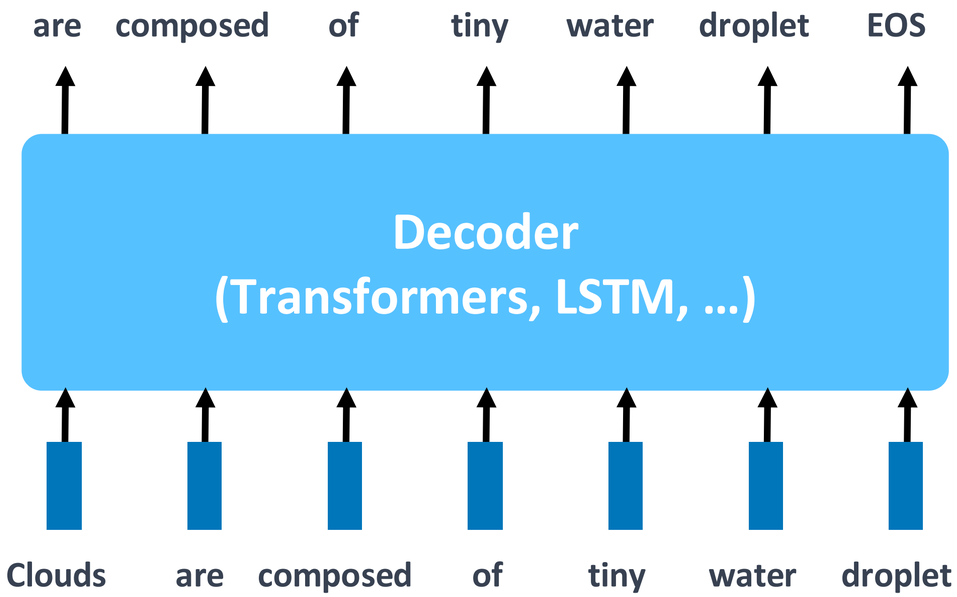
- Pre-training through **language modeling** [Dai and Le, 2015]
  - Model $P_\theta(w_t|w_{1:t-1})$, the probability distribution of the next word given previous contexts.
  - **There's lots of (English) data for this!** E.g., books, websites.
  - **Self-supervised** training of a neural network to perform the language modeling task with massive raw text data.
  - Save the network parameters to reuse later.

Why is this called self-supervised?
The labels come from the input data itself!



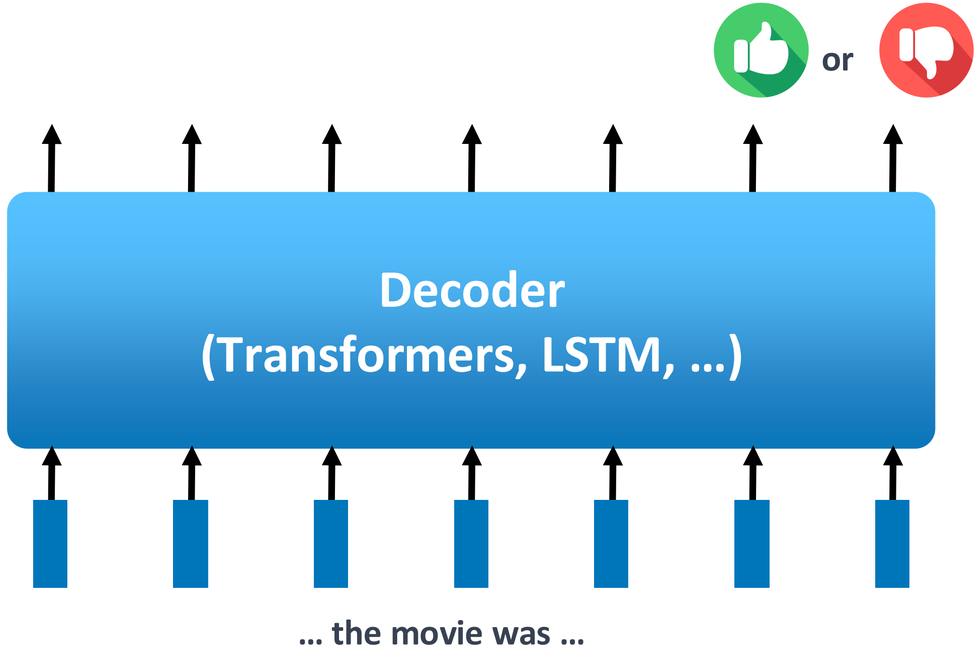
Supervised Fine-tuning for Specific Tasks

Step 1:
Pre-training



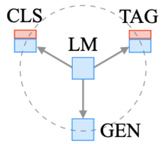
Abundant data; learn general language

Step 2:
Fine-tuning



Limited data; adapt to the task

Remember this is paradigm 3 from before

c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	

Why does this work?

Lots of Information in Raw Texts

The dish was a symphony of flavors, with each bite delivering a harmonious blend of sweet and savory notes that left my taste buds in a state of culinary _____. euphoria

The dish fell short of expectations, as the flavors lacked depth and the texture was disappointingly bland, leaving me with a sense of culinary _____ letdown

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____ disappointing

Despite a promising premise, the movie failed to live up to its potential, as the plot felt disjointed, the characters lacked depth, and the pacing left me disengaged, resulting in a rather _____ cinematic experier amazing

Lots of Information in Raw Texts

Verb	I went to Hawaii for snorkeling, hiking, and whale _____ watching
Preposition	I walked across the street, checking for traffic _____ over y shoulders.
Commonsense	I use ____ knife __ and fork to eat steak.
Time	Ruth Bader Ginsburg was born in _____ 1933
Location	University of Washington is located at _____ Seattle Washington.
Math	I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____. 34
Chemistry	Sugar is composed of carbon, hydrogen, and _____ oxygen
…	

The Stochastic Gradient Descent Angle

Why should pre-training and then fine-tuning help?

- Providing parameters $\hat{\theta}$ by approximating the pre-training loss, $min_{\theta}\mathcal{L}_{pretrain}(\theta)$.
- Then, starting with parameters $\hat{\theta}$, approximating fine-tuning loss, $min_{\theta}\mathcal{L}_{finetune}(\theta)$.
- **Stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during fine-tuning.**
  - So, maybe the fine-tuning local minima near $\hat{\theta}$ tend to generalize well!
  - And/or, maybe the gradients of fine-tuning loss near $\hat{\theta}$ propagate nicely!

Advantages of Pre-training & Fine-tuning

- **Leveraging rich underlying information** from abundant raw texts.
- **Reducing the reliance of task-specific labeled data** that is difficult or costly to obtain.
- **Initializing model parameters** for more **generalizable** NLP applications.
- **Saving training cost** by providing a reusable model checkpoints.
- **Providing robust representation** of language contexts.

Lecture Outline

1. Motivating Pre-training, aka Self-supervised Learning
2. Pre-training Architectures and Training Objectives
    1. Encoders
    2. Encoder-Decoders
    3. Decoder

3 Pre-training Paradigms/Architectures

Encoder

- E.g., BERT, RoBERTa, DeBERTa, …
- **Autoencoder** model
- **Masked** language modeling

Encoder-Decoder

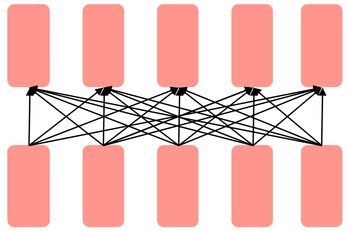
- E.g., T5, BART, …
- **seq2seq** model

Decoder

- E.g., GPT, GPT2, GPT3, …
- **Autoregressive** model
- **Left-to-right** language modeling

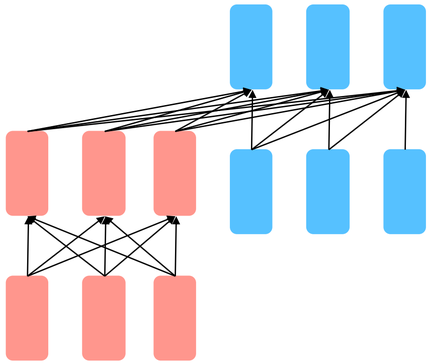
3 Pre-training Paradigms/Architectures

Encoder



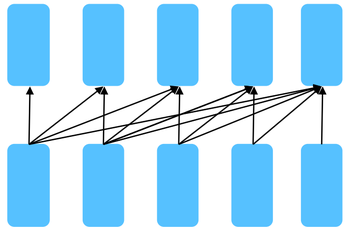
- Bidirectional; can condition on the future context

Encoder-Decoder



- Map two sequences of different length together

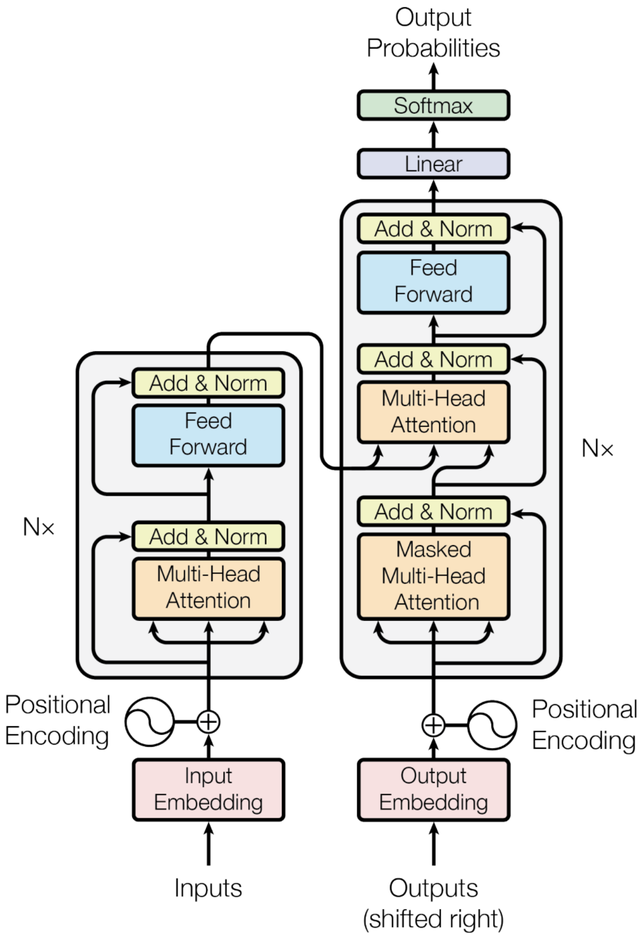
Decoder



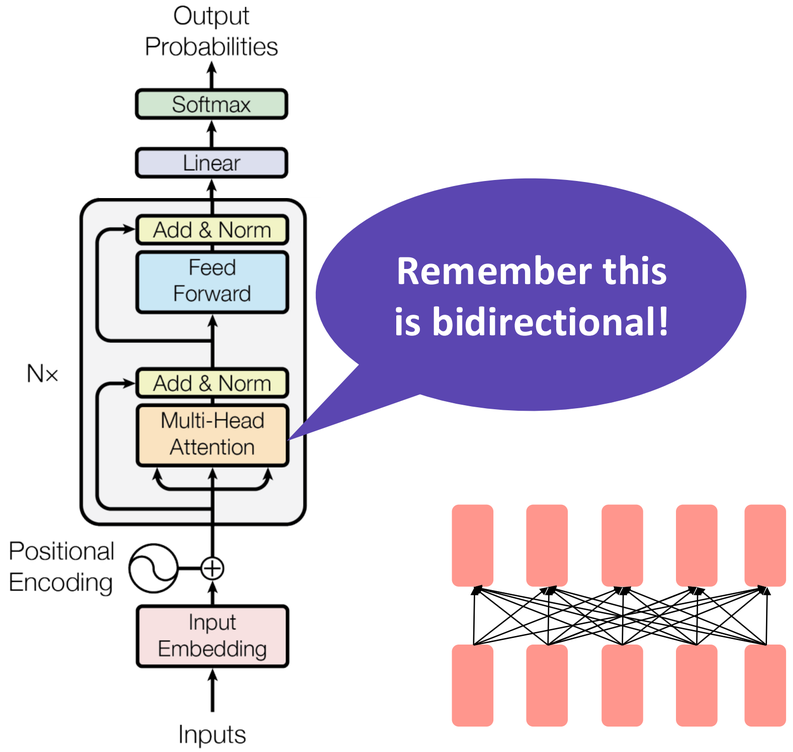
- Language modeling; can only condition on the past context

Encoder: Architecture

Full-Transformer Architecture (Encoder-Decoder)

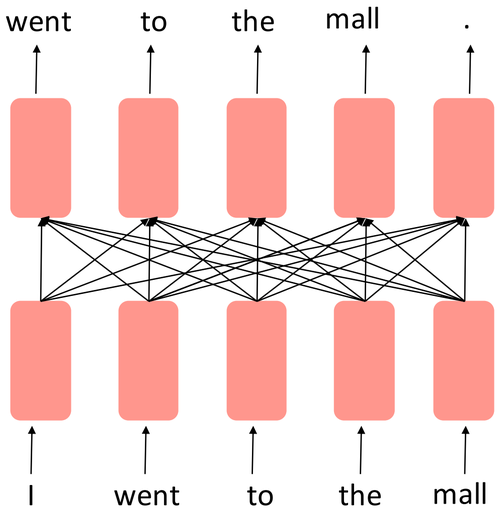


Encoder-Only Transformer Architecture



Encoder: Training Objective

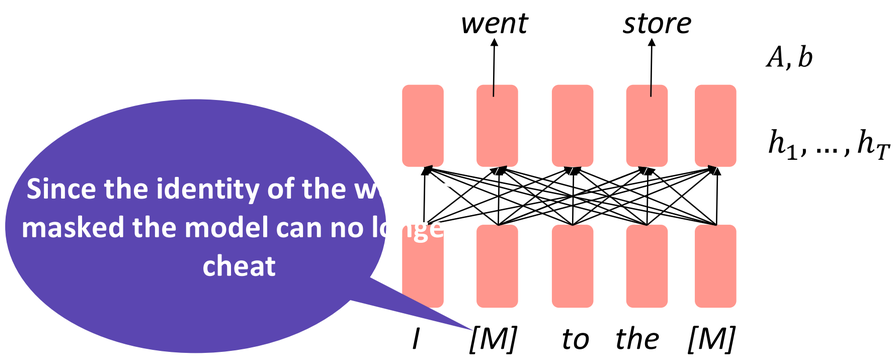
- So far, we've looked at language modeling for pre-training.
- Language Model Pretraining is problematic for encoders
- Why?
  - **Encoders get bidirectional contexts**
  - The model can cheat by just looking at the next token when predicting it without actually learning anything about language!



Encoder: Training Objective

[Devlin et al., 2018]

- How to encode information from both **bidirectional** contexts?
- General Idea: **text reconstruction!**
  - Your time is [limited don't [MA waste ving someone else's life. Don't be trapped bv [MASK], wh dogma ASK] with t living ts of other [MASK]'s thinkir people ASK] Jobs Steve



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

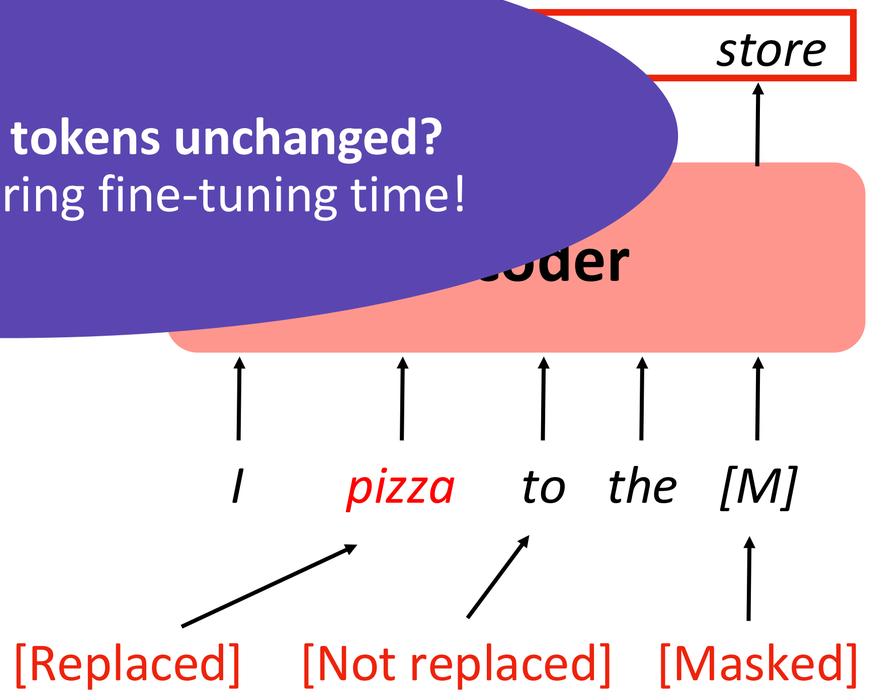
Only add loss terms from the masked tokens. If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$. Called **Masked Language model (MLM)**.

Encoder: BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers [Devlin et al., 2018]

- **2 Pre-training Objectives:**
  - **Masked LM: Choose a random 15% of tokens to predict**
    - For each chosen token:
      - Replace it with **[MASK]** 80
      - Replace it with a **random t**
      - Leave it **unchanged** 10% of the t
  - **Next Sentence Prediction (NSP)**
    - 50% of the time two adjacent sentences are in the correct order.
    - **This actually hurts model learning based on later work!**

WHY keeping some tokens unchanged?
There's no [MASK] during fine-tuning time!



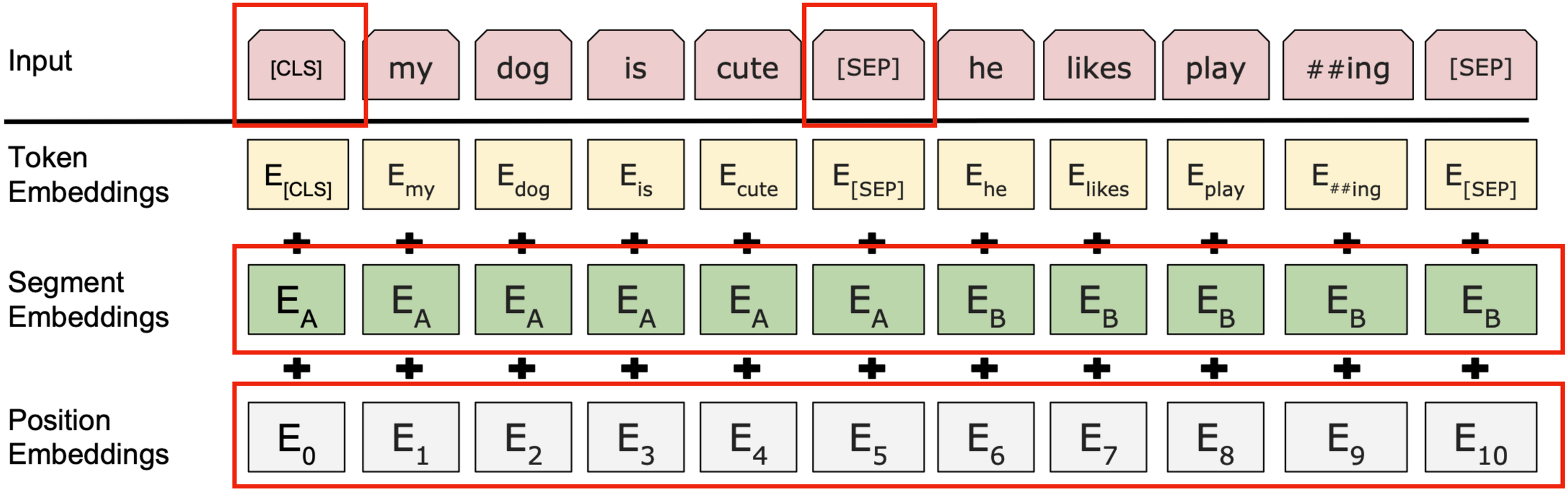
Encoder: BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers [Devlin et al., 2018]

Special token added to the beginning of each input sequence

Special token to separate sentence A/B

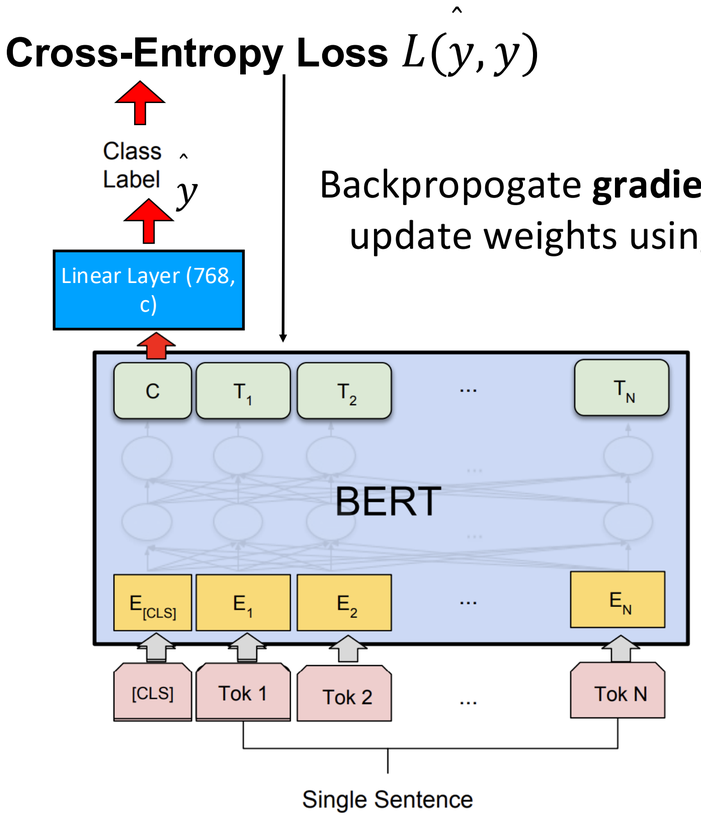
Final embedding is the sum of all three!



Learned embedding to every token indicating whether it belongs to sentence A or sentence B

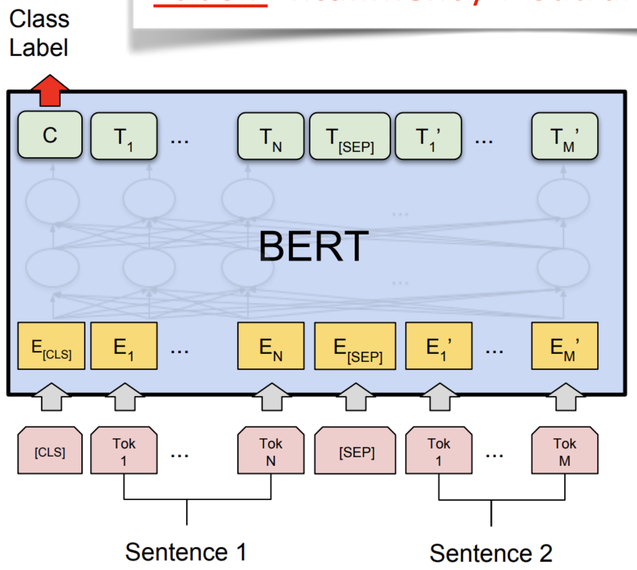
Position of the token in the entire sequence

Encoder: BERT (Fine-tuning)



Single-Sentence Tasks like SST-2 (Sentiment Analysis)

Input:
Premise: A soccer game with multiple males playing
Hypothesis: Some men are playing a sport
Label: **Entailment** / Neutral / Contadiction



Sentence Pair Classification Tasks like Natural Language Inference

Encoder: BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers [Devlin et al., 2018]

- **SOTA at the time on a wide range of tasks after fine-tuning!**

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
$\text{BERT}_{\text{BASE}}$	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**

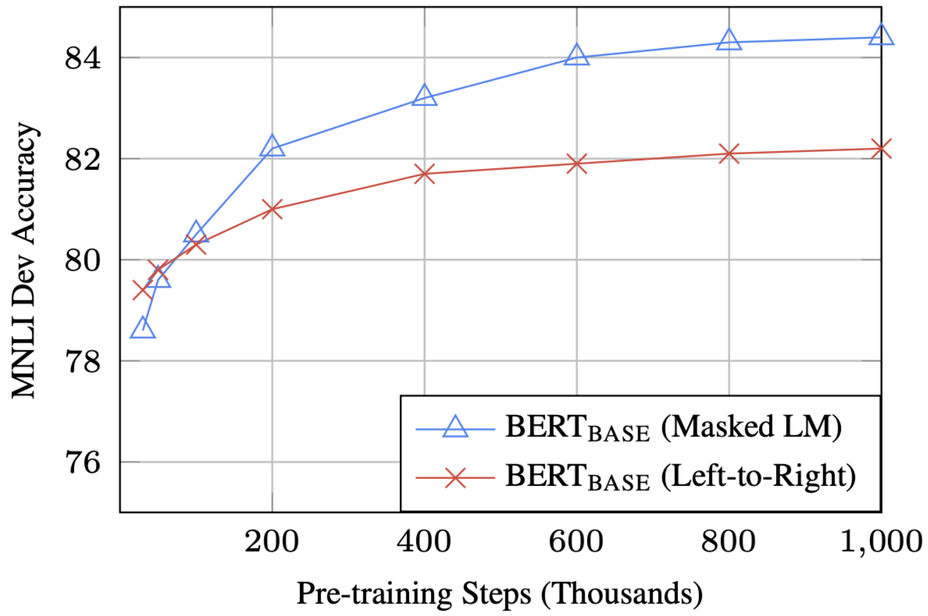
- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI:** natural language inference over question answering data
- **SST-2:** sentiment analysis
- **CoLA:** corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B:** semantic textual similarity
- **MRPC:**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

Encoder: BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers [Devlin et al., 2018]

SWAG (Commonsense inference task)

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0



- **Two Sizes of Models**
  - **Base:** 110M, 4 Cloud TPUs, 4 days
  - **Large:** 340M, 16 Cloud TPUs, 4 days
  - Both models can be fine-tuned with single GPU
  - The larger the better!

- MLM converges slower than Left-to-Right at the beginning, but out-performers it eventually

Encoder: RoBERTa

[Liu et al., 2019]

- **Original BERT is significantly undertrained!**
- More data (16G => 160G)
- Pre-train for longer
- Bigger batches
- Removing the next sentence prediction (NSP) objective
- Training on longer sequences
- Dynamic masking, randomly masking out different tokens
- A larger byte-level BPE vocabulary containing 50K sub-word units

All around better than BERT!

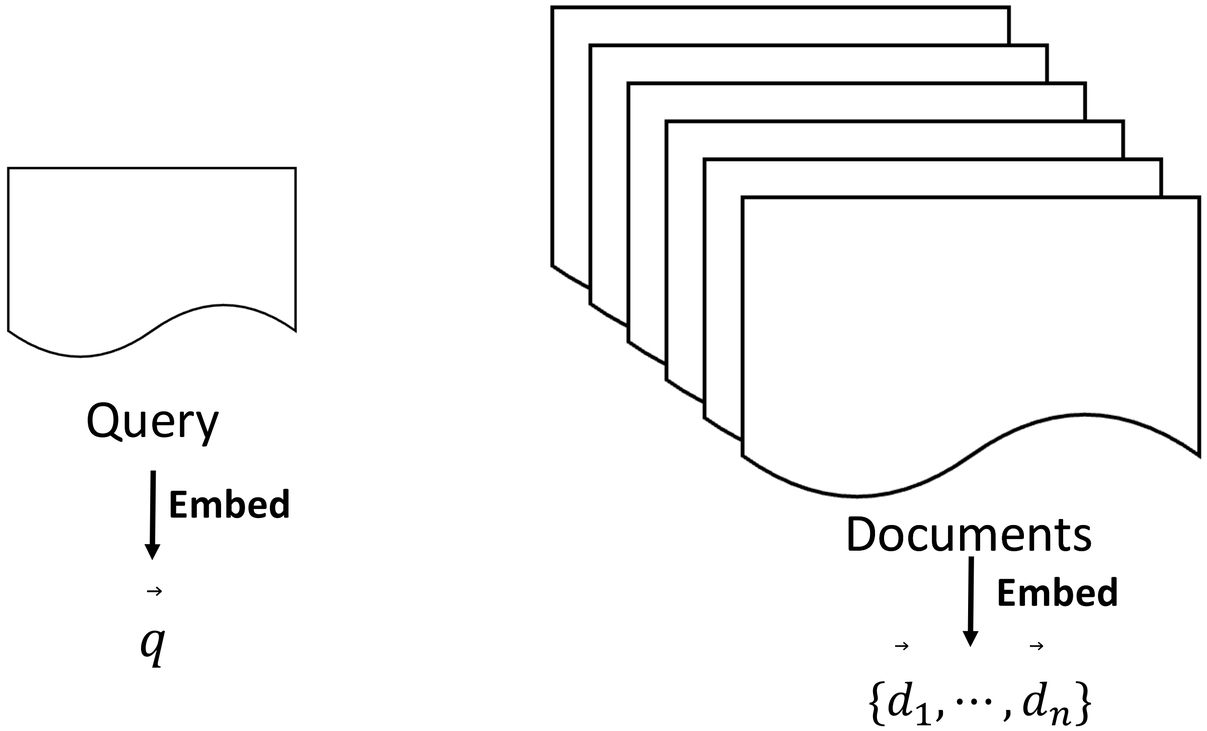
Encoder: Other Variations of BERT

- **ALBERT** [Lan et al., 2020]: incorporates two parameter reduction techniques that lift the major obstacles in scaling pre-trained models
- **DeBERTa [He et al., 2021]:** decoding-enhanced BERT with disentangled attention
- **SpanBERT** [Joshi et al., 2019]**:** masking contiguous spans of words makes a harder, more useful pre-training task
- **ELECTRA** [Clark et al., 2020]**:** corrupts texts by replacing some tokens with plausible alternatives sampled from a small generator network, then train a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample or not.
- **DistilBERT** [Sanh et al., 2019]**:** distilled version of BERT that's 40% smaller
- **TinyBERT** [Jiao et al., 2019]**:** distill BERT for both pre-training & fine-tuning
- **…**

Encoders for Information Retrieval

Retrieve the set of relevant documents given a query

Applications:
- **Search Engines (This is how google works!)**
- **Retrieval Augmented Language Models**



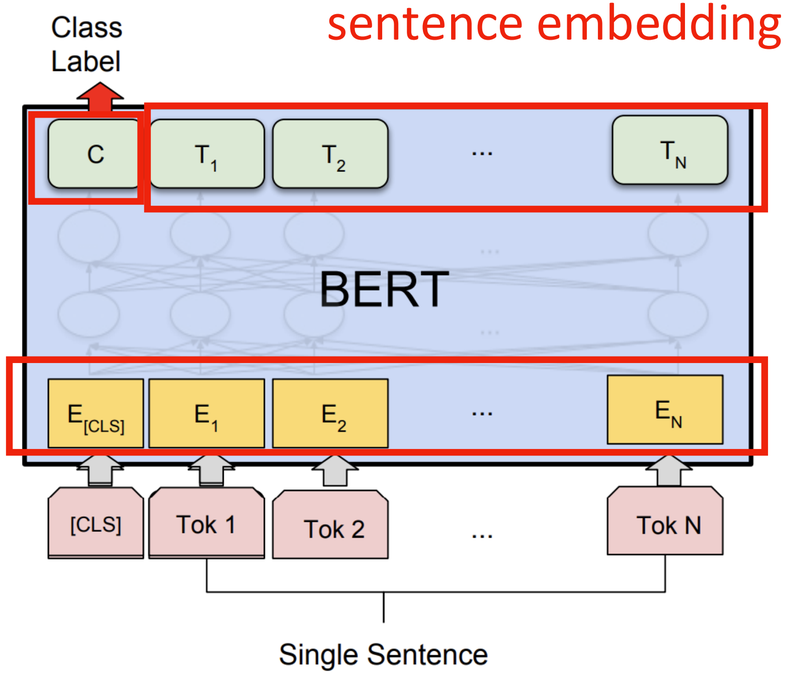
Score document relevance by, e.g., computing cosine similarity between the query and the document

$$\textit{relevance}-\textit{score}(d \mid q) = \cos(\hat{q}, \hat{d})$$

Encoders for Information Retrieval

How do we get sentence embeddings from an encoder-based model like BERT?

Option 3: Use representations of CLS token for sentence embedding



Option 2: Average learned **contexual** word embeddings

Option 1: Average learned word embeddings

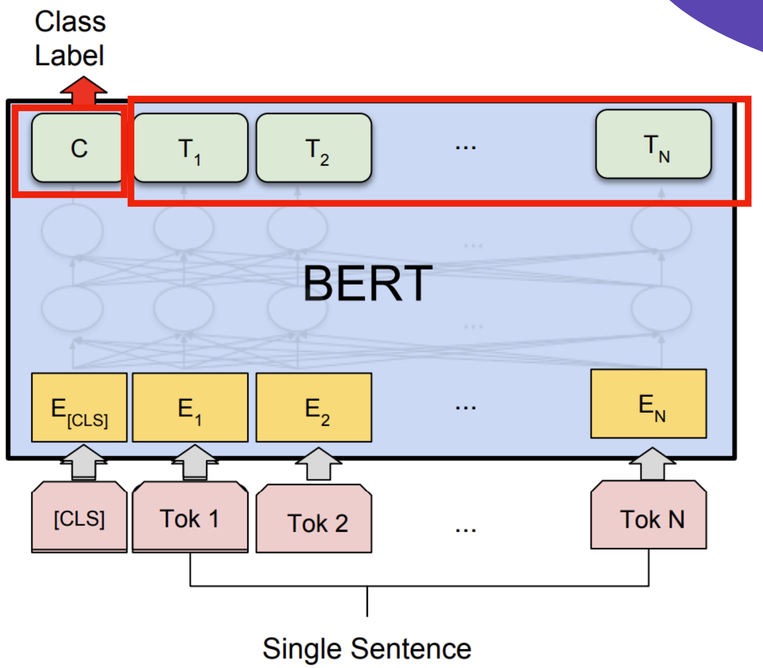
Problem: Representations not contextual! Equivalent to using GloVe vectors

Encoders for Information Retrieval

Out of the box even e… very good f…

Why?
Pre-training objective unlike for word2vec is not aligned with the objective of placing similar sentences closer in embedding space

Performance is even worse than averaging word embeddings!



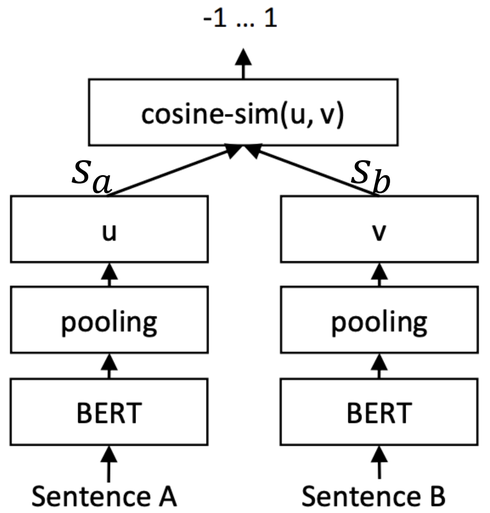
Model	STS12	STS13	STS14	STS15	STS16	STSb	S…R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.7…	61.32
Option 2 Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
Option 3 BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22

Spearman correlations for Textual Similarity (STS) tasks
(higher is better)

Encoders for Information Retrieval: Sentence BERT (S-BERT)

- Finetune BERT / RoBERTa to learn sentence level representations such that similar sentences are located closer in the embedding space

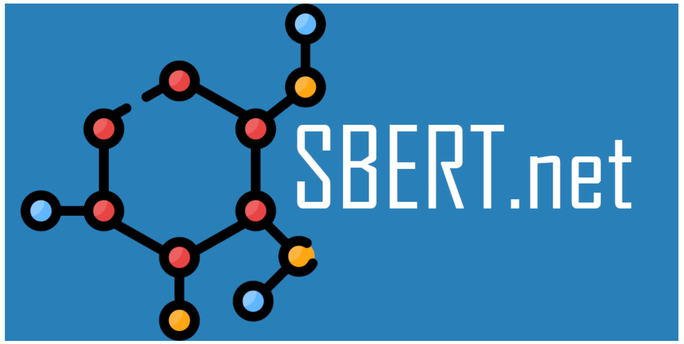
- Uses a triplet objective function — Given an anchor sentence a, a positive sentence p, and a negative sentence n, triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n.



Triplet objective function

$$max(\| s_a - s_p \| - \| s_a - s_n \| + \epsilon, 0)$$

Encoders for Information Retrieval: Sentence BERT (S-BERT)



Sentence Transformers Library. Very handy for using pre-trained Sentence-BERT-like models

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

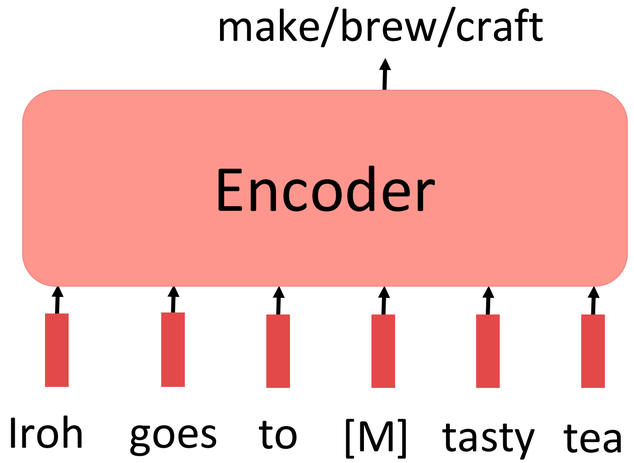
Spearman correlations for Textual Similarity (STS) tasks
(higher is better)

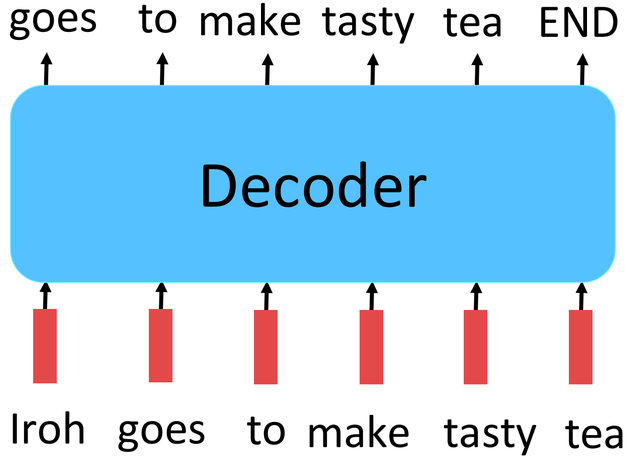
Sentence-BERT / RoBERTa performs remarkably better than the existing approaches!

Encoder: Pros & Cons

- Consider both left and right context
- Capture intricate contextual relationships

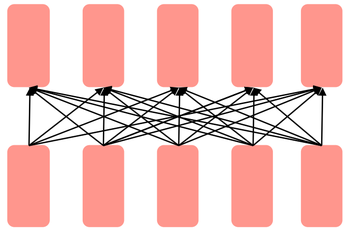
- Not good at generating open-text from left-to-right, one token at a time





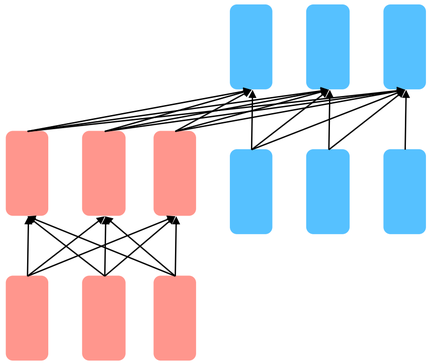
3 Pre-training Paradigms/Architectures

Encoder



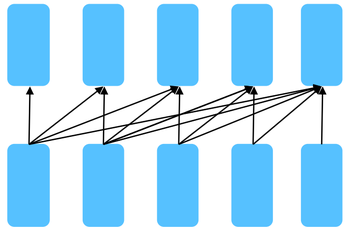
- Bidirectional; can condition on the future context

Encoder-Decoder



- Map two sequences of different length together

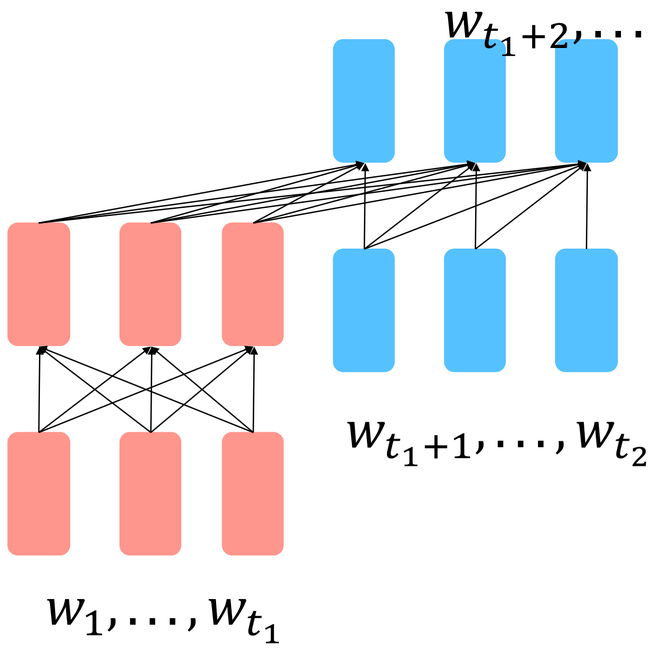
Decoder



- Language modeling; can only condition on the past context

Encoder-Decoder: Architecture

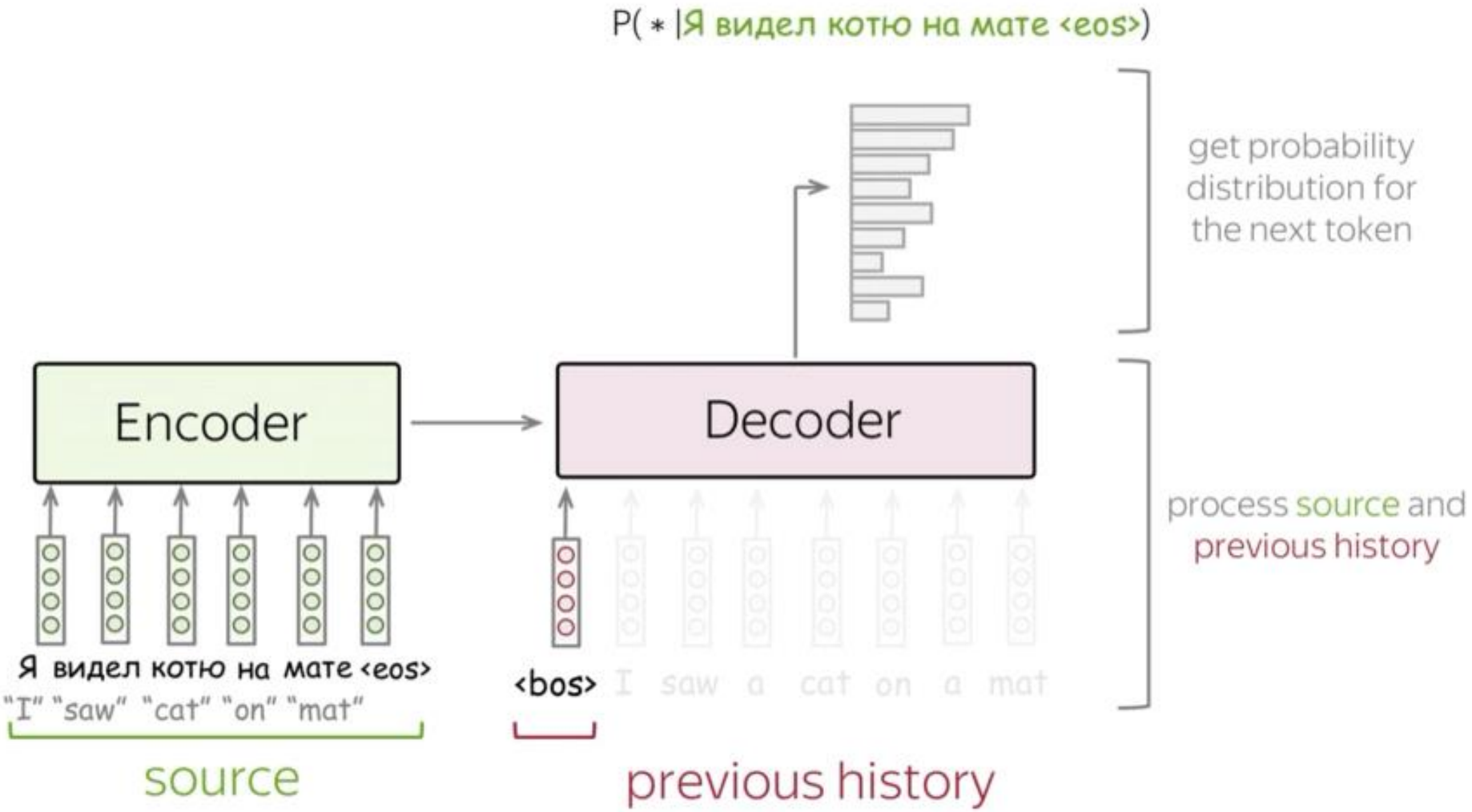
- Moving towards **open-text generation**…
- **Encoder** builds a representation of the source and gives it to the **decoder**
- **Decoder** uses the source representation to generate the target sentence
- The **encoder** portion benefits from **bidirectional** context; the **decoder** portion is used to train the whole model through **language modeling**



$$h_1, \ldots, h_{t_1} = Encoder(w_1, \ldots, w_{t_1})$$
$$h_{t_1+1}, \ldots, h_{t_2} = Decoder(w_{t_1+1}, \ldots, w_{t_2}, h_1, \ldots, h_{t_1})$$
$$y_i \sim Ah_i + b, i > t$$

[Raffel et al., 2018]

Encoder-Decoder: An Machine Translation Example



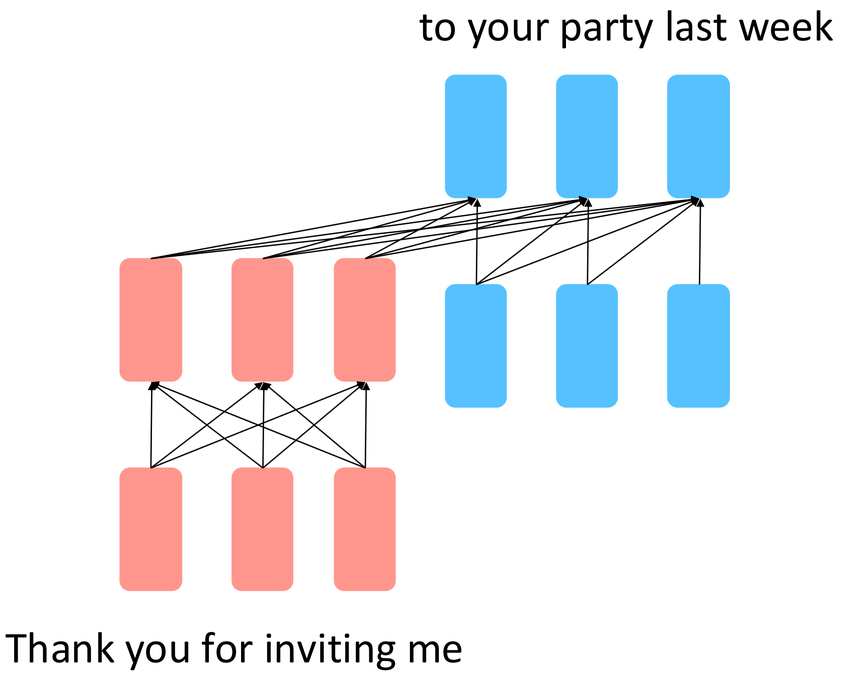
[Lena Viota Blog]

Encoder-Decoder: Training Objective

- Can we use Language Modeling here?
- **<u>Kinda</u>:** Given a text span, choose a random point to split it into prefix and target portions.
  - Encoder takes the prefix as input and the decoder is trained to generate the target given prefix

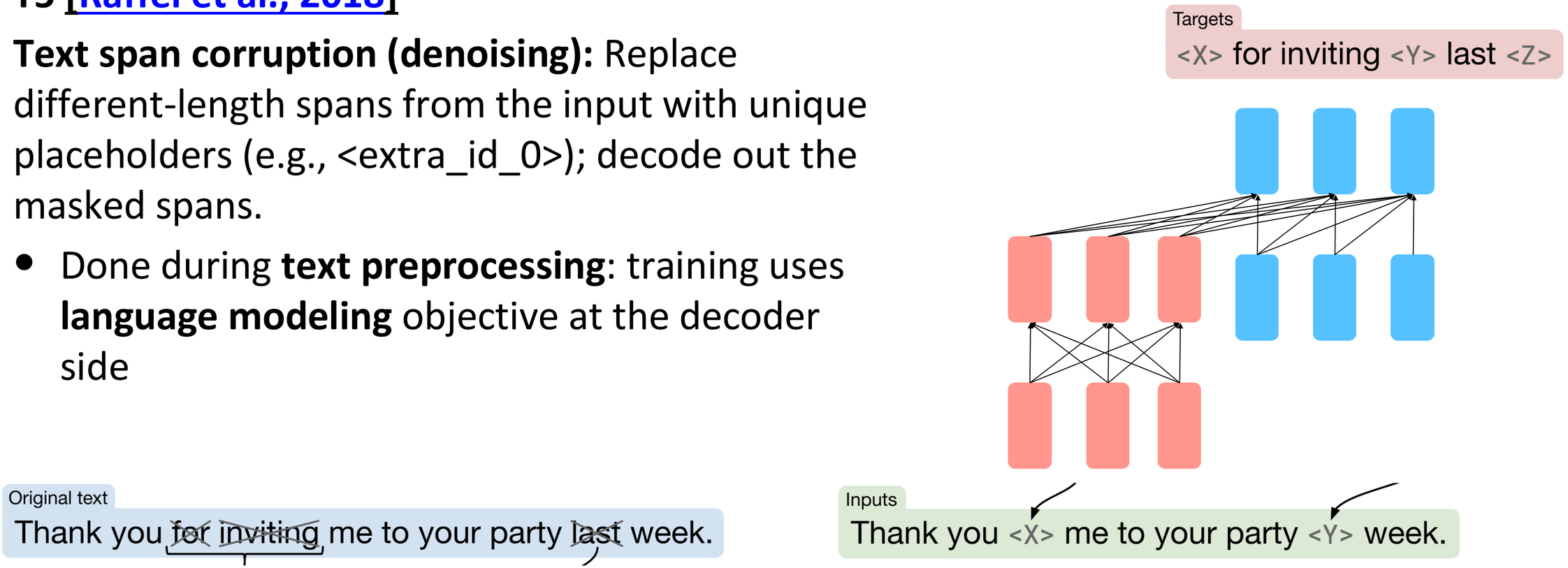
Thank you for inviting me to your party last week.

e.g. split here



Encoder-Decoder: Training Objective

- **T5 [Raffel et al., 2018]**
- **Text span corruption (denoising):** Replace different-length spans from the input with unique placeholders (e.g., <extra_id_0>); decode out the masked spans.
  - Done during **text preprocessing**: training uses **language modeling** objective at the decoder side



Encoder-Decoder: T5

Text **t**o **T**ext **T**ransfer **T**ransformer [Raffel et al., 2019]

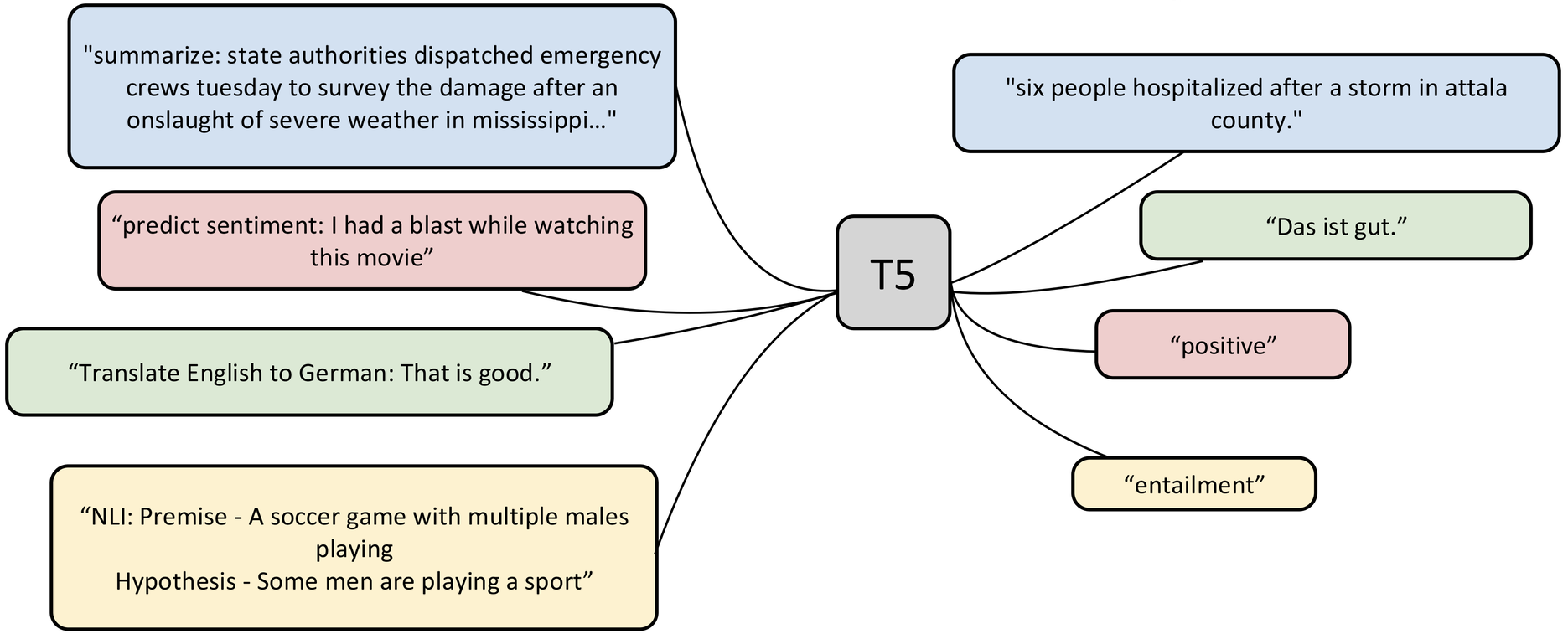
- **Span corruption (denoising)** objective works better than language modeling
- **Encoder-decoders** works better than decoders

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

Decoder (coming next!)

Encoder-Decoder: T5 (Fine-tuning)

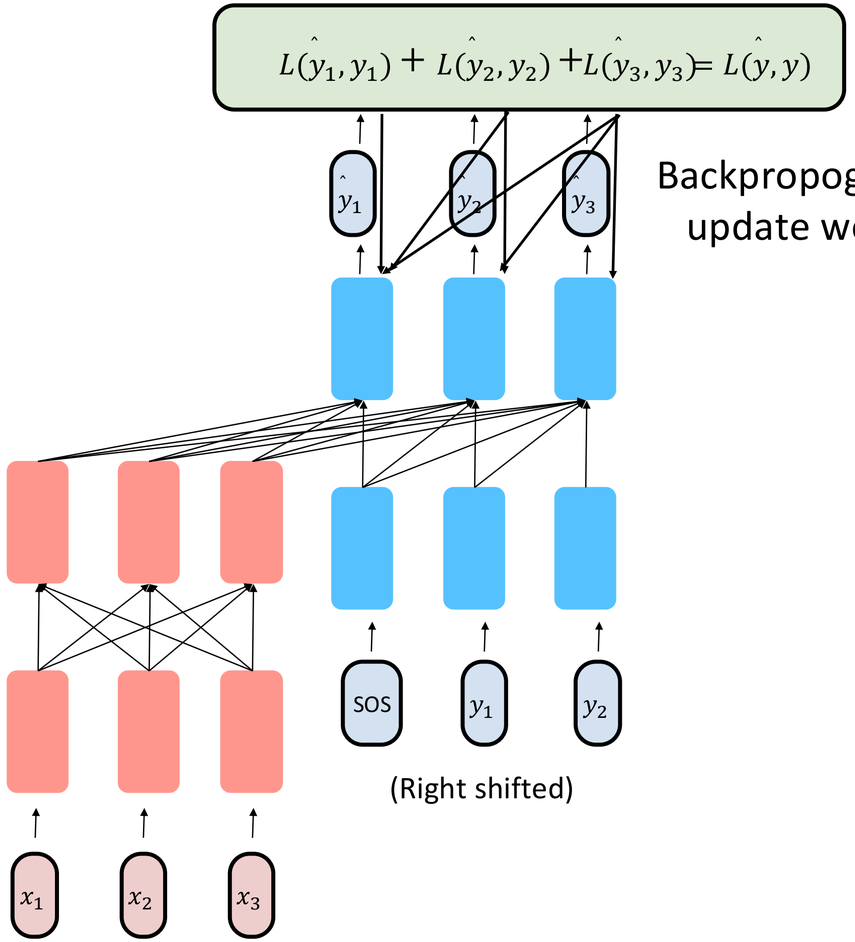
Core Idea: Cast any NLP task at hand as a **text generation problem** given some input text!



Encoder-Decoder: T5 (Fine-tuning)

X “Translate English to German: That is good.”

Y “Das ist gut.”



Backpropogate **gradients** and update weights using **SGD**

Encoder-Decoder: T5

- **Text-to-Text:** convert NLP tasks into input/output text sequences
- **Dataset:** Colossal Clean Crawled Corpus (C4), 750G text data!
- **Various Sized Models:**
  - Base (222M)
  - Small (60M)
  - Large (770M)
  - 3B
  - 11B
- **Achieved SOTA with scaling & pu**

[Google Blog]

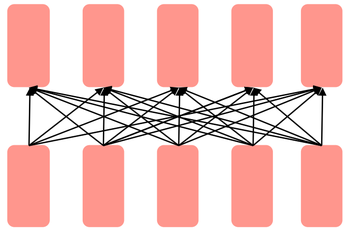
Encoder-Decoder: Pros & Cons

- A nice middle ground between leveraging **bidirectional** contexts and **open-text** generation
- Good for **multi-task** fine-tuning

- Require more **text wrangling**
- **Harder to train**
- **Less flexible** for natural language generation

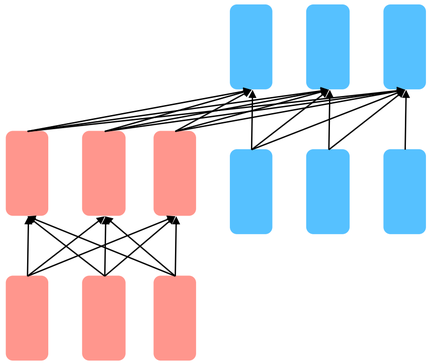
3 Pre-training Paradigms/Architectures

Encoder



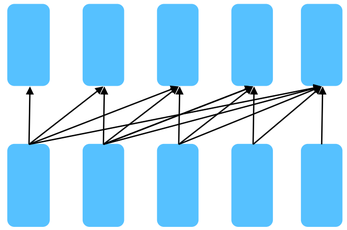
- Bidirectional; can condition on the future context

Encoder-Decoder



- Map two sequences of different length together

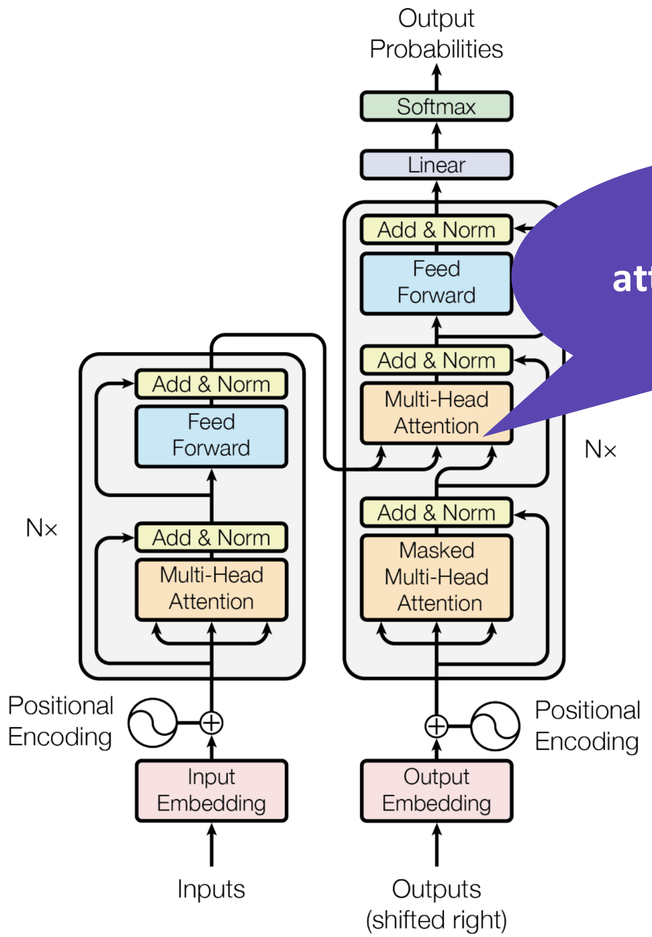
Decoder



- Language modeling; can only condition on the past context

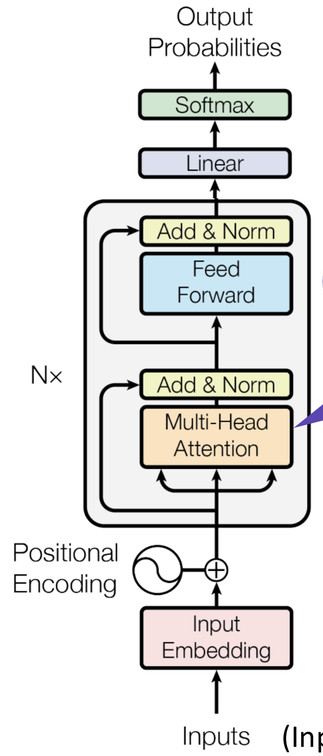
Decoder: Architecture

Full-Transformer Architecture (Encoder-Decoder)

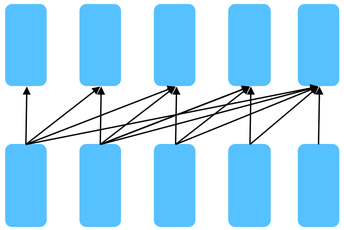


Get rid of cross attention as we do not have an encoder!

Decoder-Only Transformer Architecture

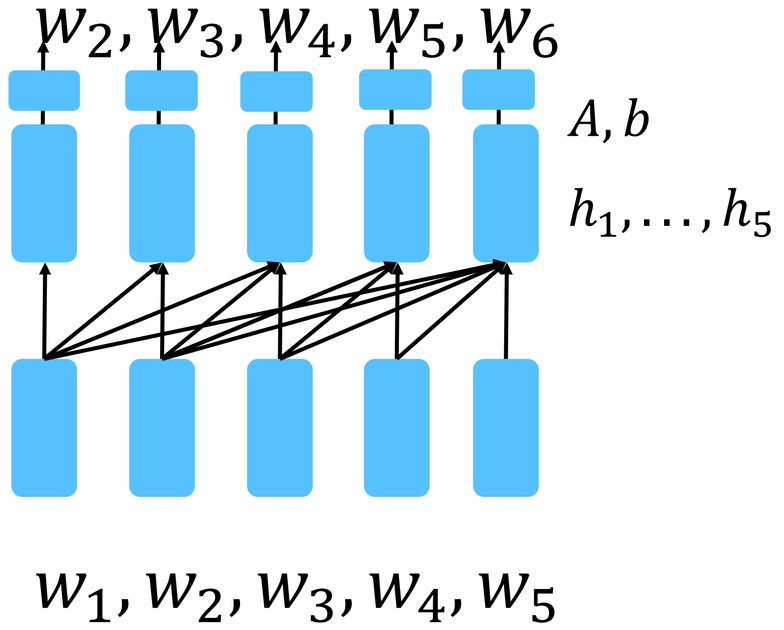


Unidirectional Attention i.e. only attends to past tokens



Decoder: Training Objective

- Many most famous generative LLMs are **decoder-only**
  - e.g., GPT1/2/3/4, Llama1/2/3, Claude, Gemini…
- **Language modeling!** Natural to be used for **open-text generation**
- **Conditional LM:** $p(w_t|w_1,\ldots,w_{t-1},x)$
  - Conditioned on a source context x to generate from left-to-right
- Can be fine-tuned for **natural language generation (NLG)** tasks, e.g., dialogue, summarization.



Decoder: GPT

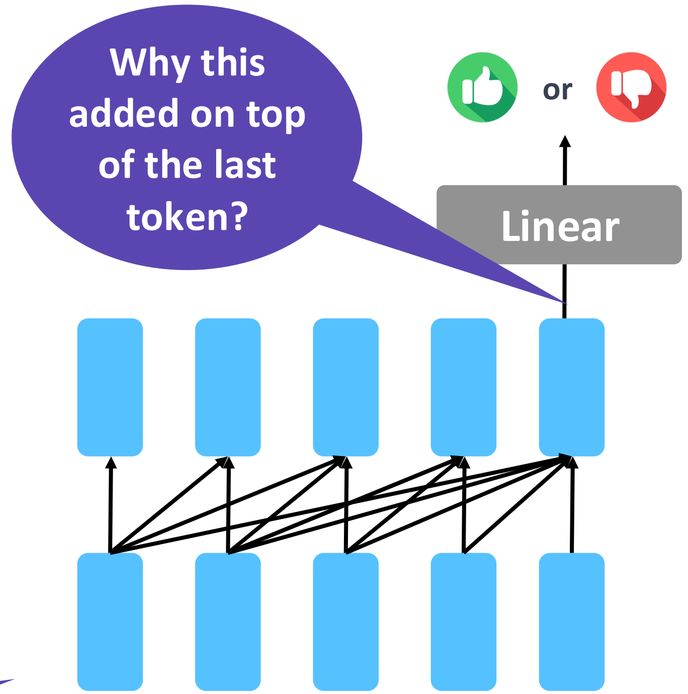
Improving Language Understanding by **G**enerative **P**re-**T**raining [Radford et al., 2018]

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

- Transformer decoder with **12 layers**, **117M parameters**.
- 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"

Decoder: GPT (Finetuning)

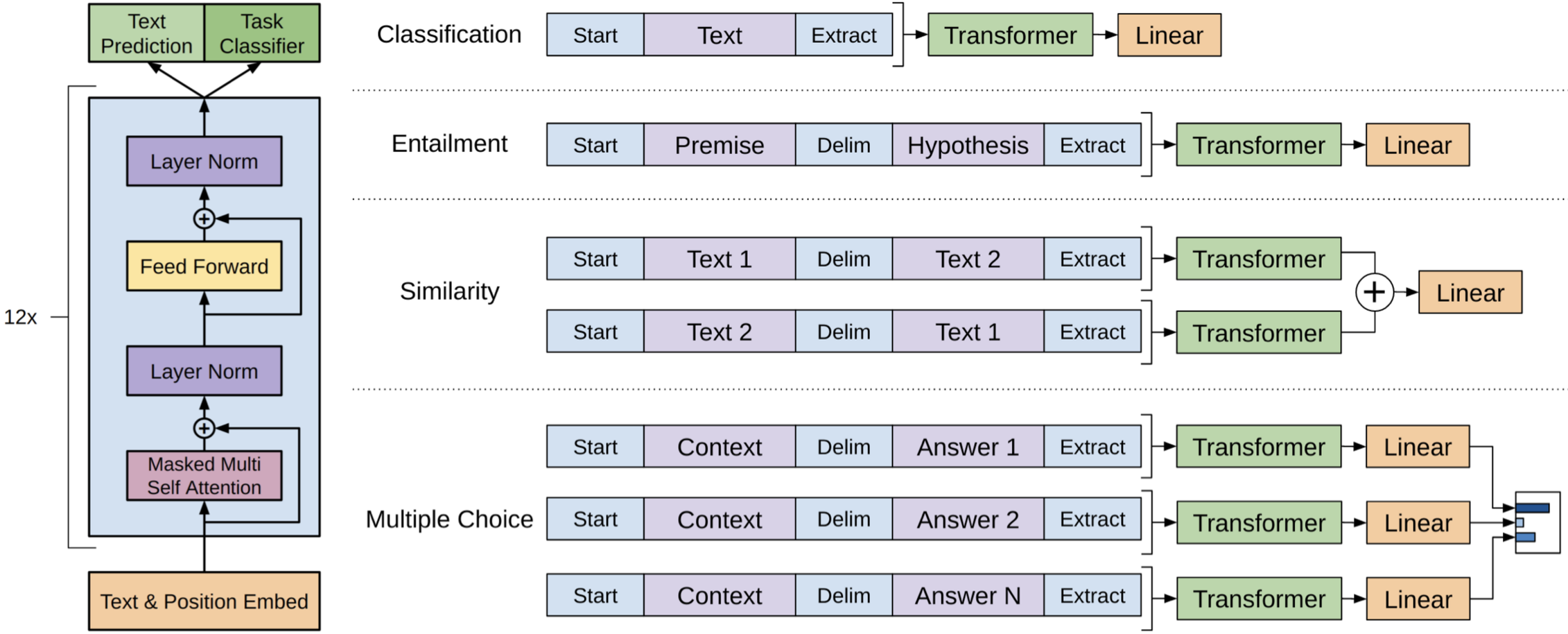
- Customizing the pre-trained model for downstream tasks:
  - Add a **linear layer** on top of the last hidden layer to make it a classifier!
  - During fine-tuning, trained the randomly **initialized linear layer**, along with **all parameters** in the neural net.



I had a blast while watching this movie.

While not originally formulated this way, we can use T5-style text-to-text fine-tuning here for any task. In fact thats the norm now!

Decoder: GPT (Finetuning)



Decoder: GPT-2

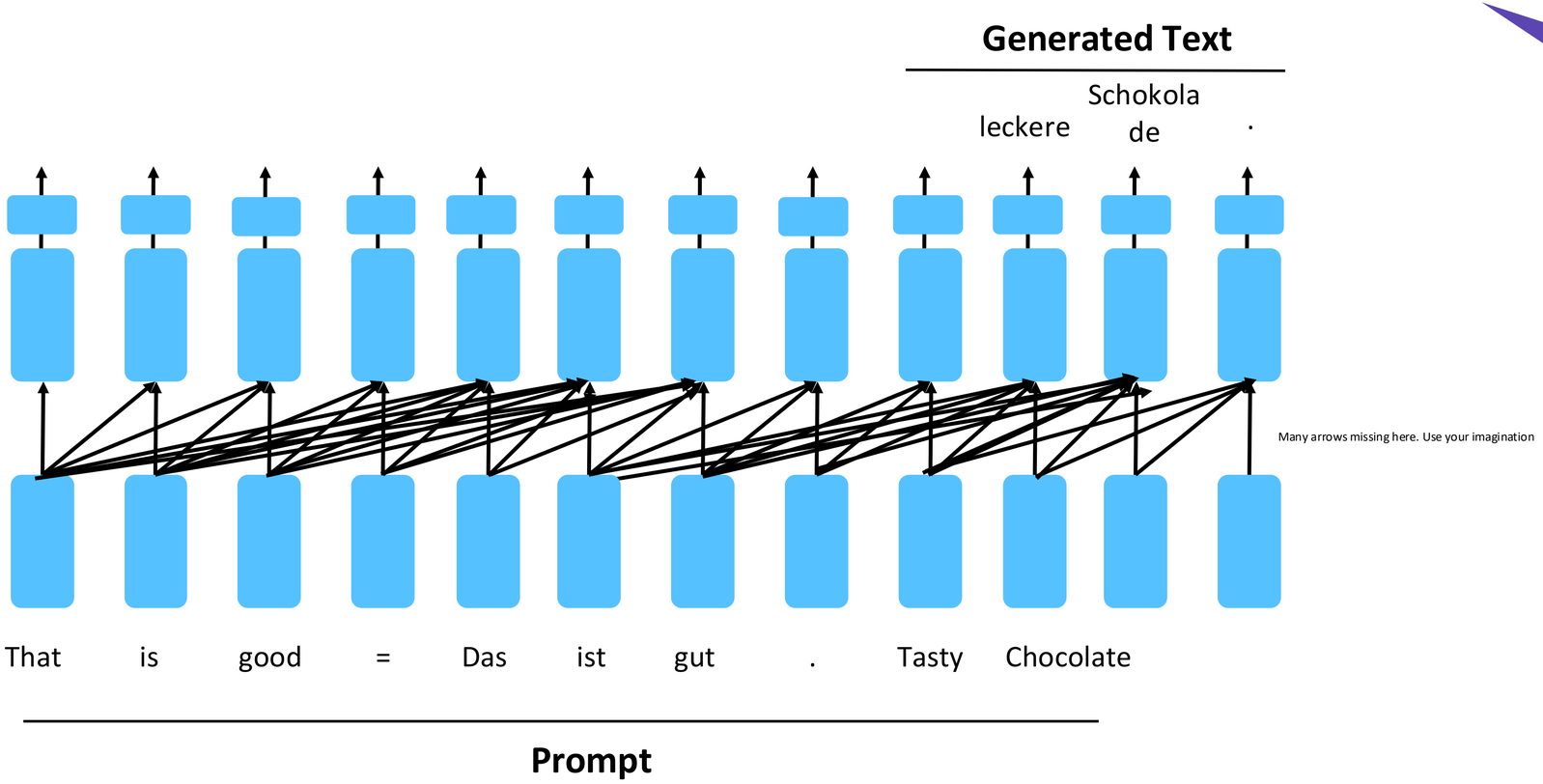
Language Models are Unsupervised Multitask Learners [Radford et al., 2019]

- Scaled-up version of GPT. The largest GPT-2 Model had **1.56B parameters** with **48 layers**.
- Was trained on a much larger dataset
  - **WebText**, curated for high-quality text
  - Consisted of web scrapes of outbound links from Reddit with at least 3 upvotes
  - 45 million links -> 8 million documents -> 40GB of text

Decoder: GPT-2

Language Models are **Unsupervised Multitask Learners** [Radford et al., 2019]

- One of the most impressive things about GPT-2 was that it could obtain great performance on many NLP datasets zero-shot!



i.e. no fine-tuning and simply prompting the pre-trained model and generating the output

Decoder: GPT-2

Language Models are **Unsupervised Multitask Learners** [Radford et al., 2019]

- One of the most impressive things about GPT-2 was that it could obtain great performance on many NLP datasets zero-shot!

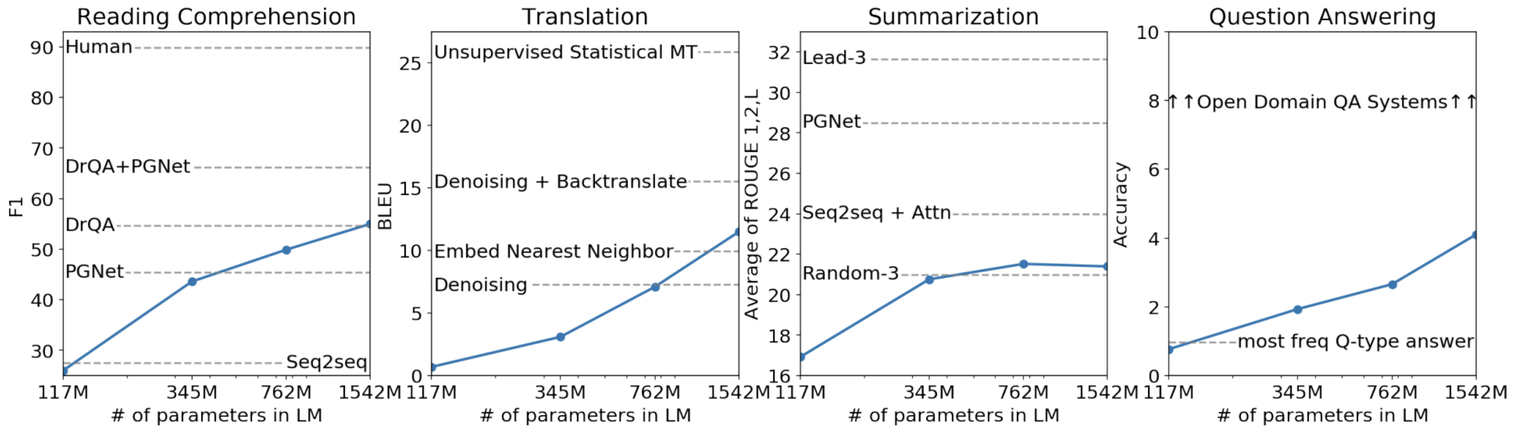


Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Would spark the beginning of Era of Prompting (Paradigm 4)

The GPT-2 paper didn't even have any fine-tuning experiments!

Decoder: GPT-2 (Text Generation)

[Radford et al., 2019]

- The model w s of natural langu

February 14, 2019

Better language models and their implications

Read paper ↗ View code ↗

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.

Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

More on text generation soon!

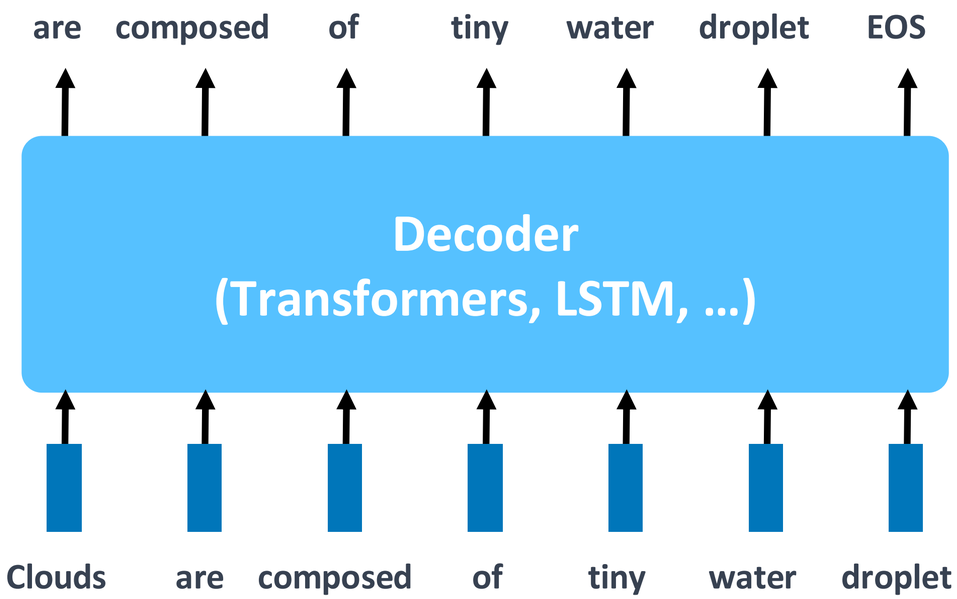
At the time of release OpenAI didn't release the 1.5B version of GPT-2 to prevent generating deceptive, biased, or abusive language at scale

How to pick a proper architecture for a given task?

- Right now **decoder-only** models seem to dominant the field at the moment
  - e.g., GPT1/2/3/4/5, Mistral, Llama1/2/3/3.1,4, Gemini, Claude….
  - Best models for text generation
- Encoders (BERT) are good if you want light-weight models for NLU-like problems or need sentence embeddings for retrieval
- T5 (seq2seq) works well with multi-tasking. Some evidence they are better for NLU than decoders [Tay et al. 2022. UL2]
- But, mostly, we all just use API models and don’t fine-tune anything ourselves…

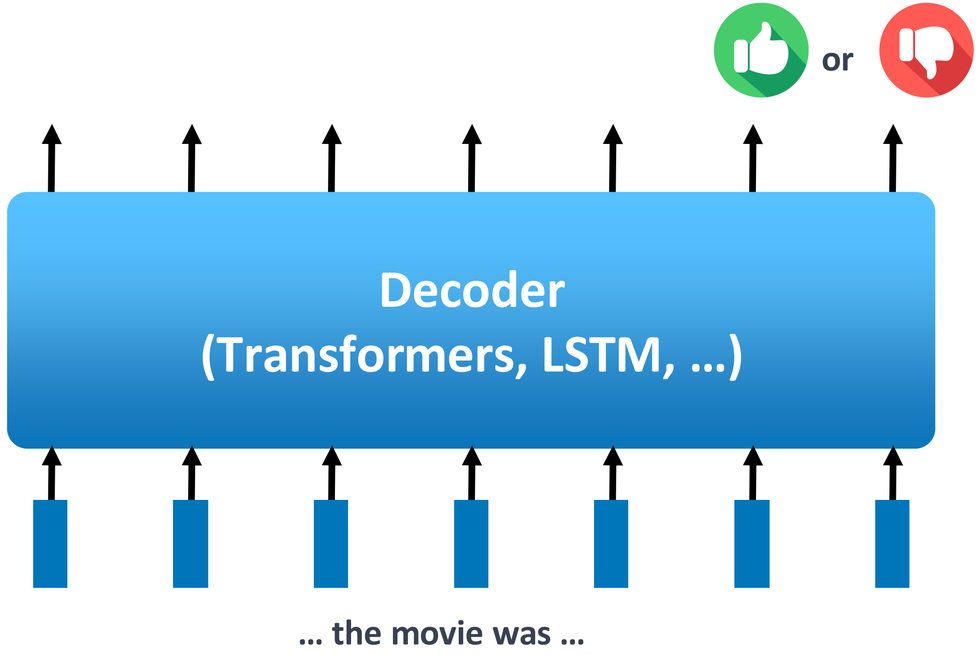
Bonus: Forgetting and Parameter Efficiency

Step 1:
Pre-training



Abundant data; learn general language

Step 2:
Fine-tuning



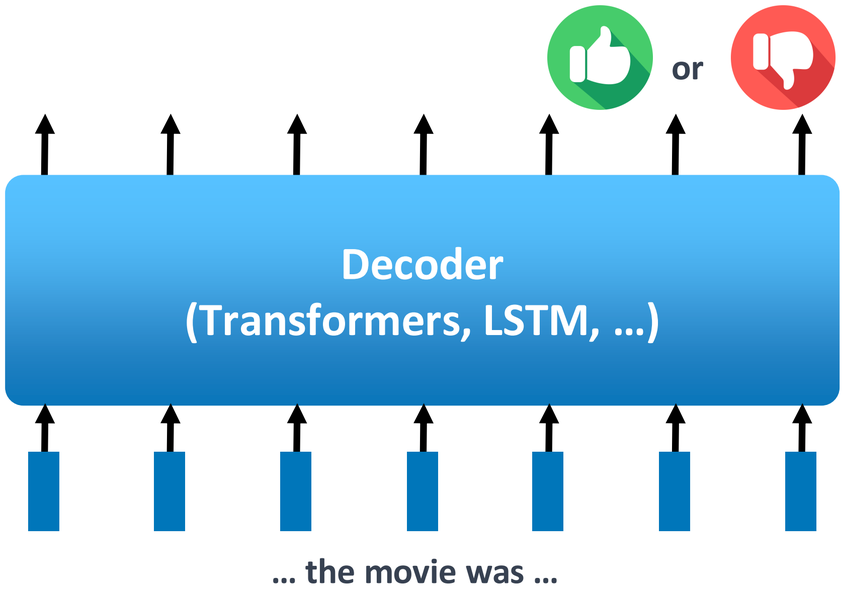
Limited data; adapt to the task

Caveat: Catastrophic Forgetting

- **Sequentially** pre-train then fine-tune may result in **catastrophic forgetting**, meaning that **while adapting to the new fine-tuning task, the model may lose previously learned information.**
- However, as modern language models are becoming larger in size and are pre-trained on massive raw text, they do encode tremendous amount of valuable information. **Thus, it's generally still more helpful to leverage information learned from the pre-training stage, than training on a task completely from scratch.**

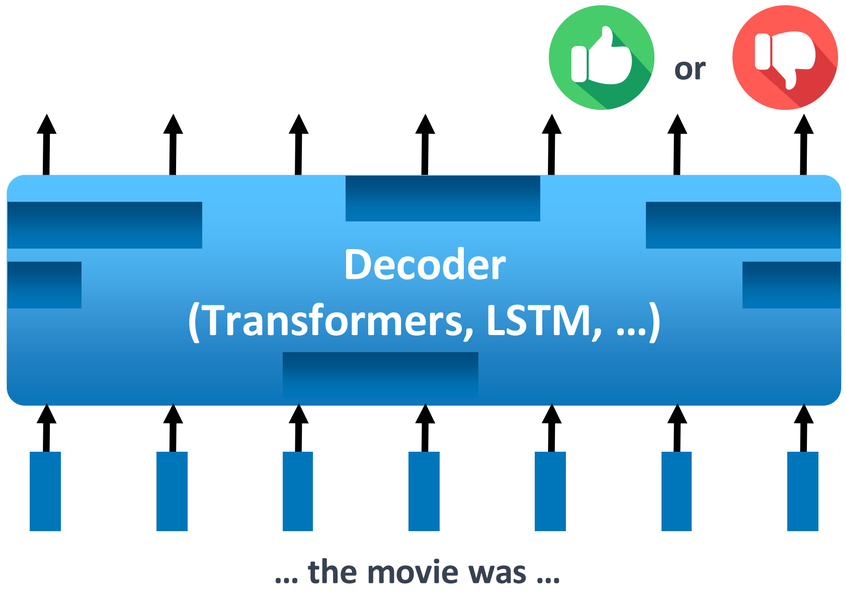
Parameter-Efficient Fine-tuning

Instead of updating all parameters in the massive neural network (up to many billions of parameters), **can we make fine-tuning more efficient?**



Full Fine-tuning

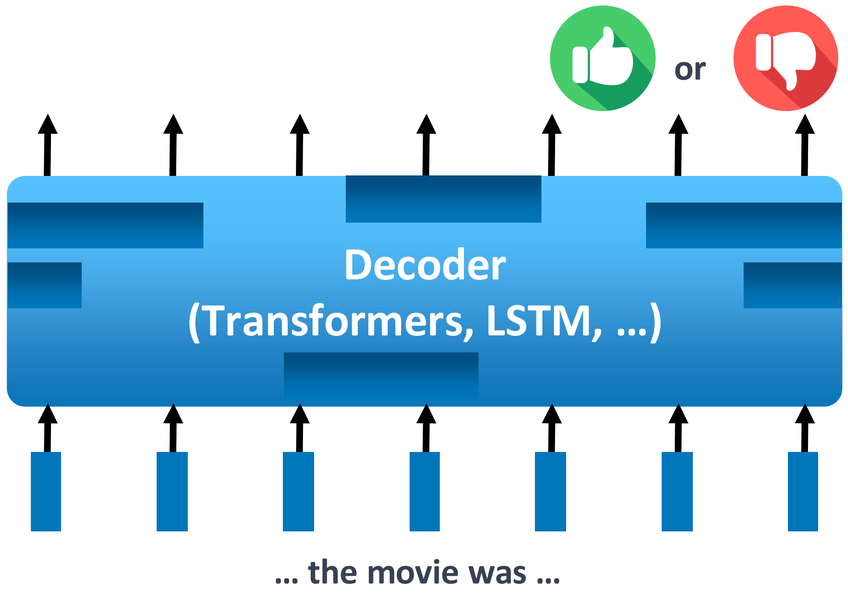
Updating all parameters



Parameter-Efficient Fine-tuning

Updating a few existing or new parameters

Parameter-Efficient Fine-tuning



Parameter-Efficient Fine-tuning

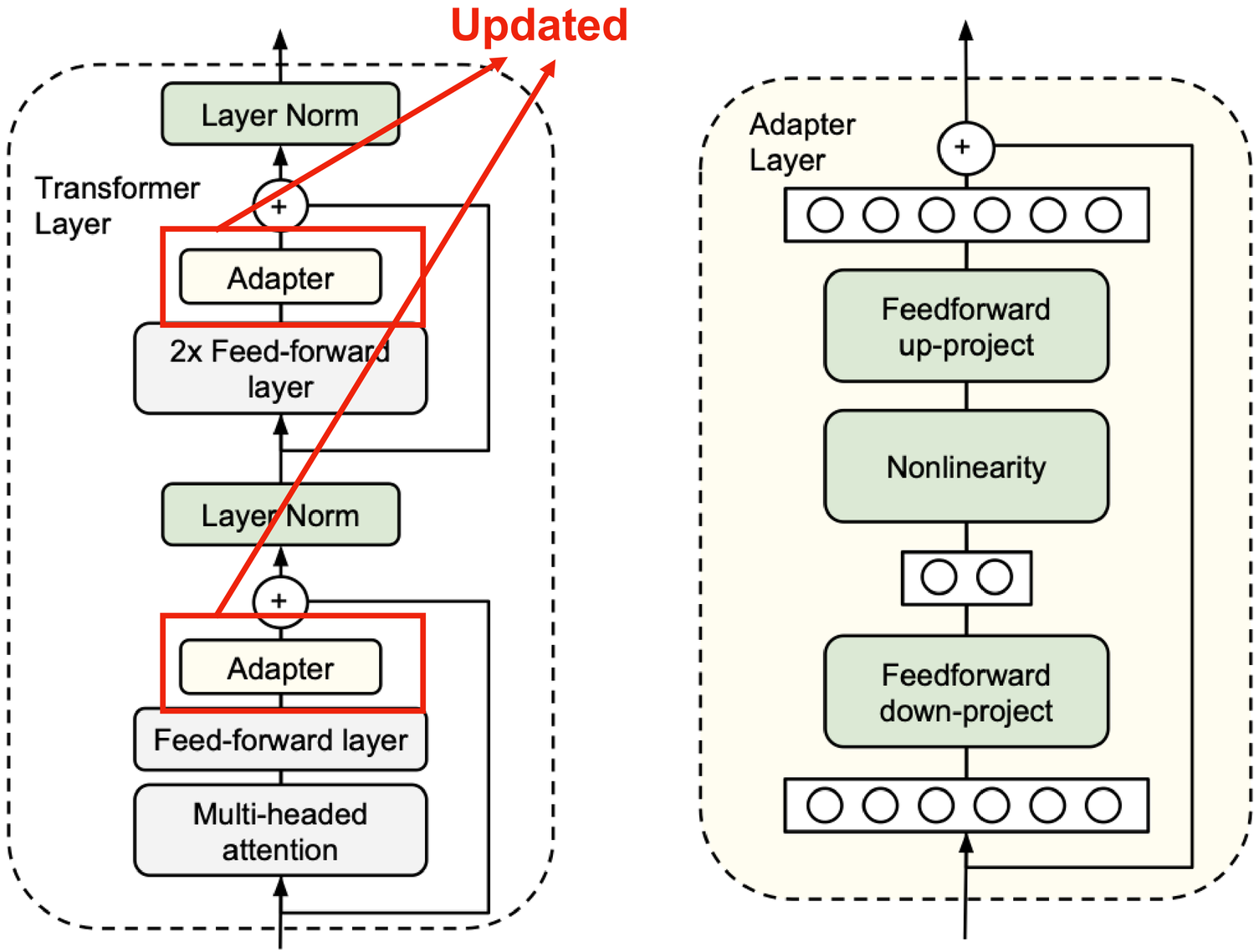
Updating a few existing or new parameters

- **More efficient at fine-tuning & inference time**
- **Less overfitting** by keeping the majority of parameters learned during pre-training

Adapter

- Injecting **new layers** (randomly initialized) into the original network, keeping other parameters frozen

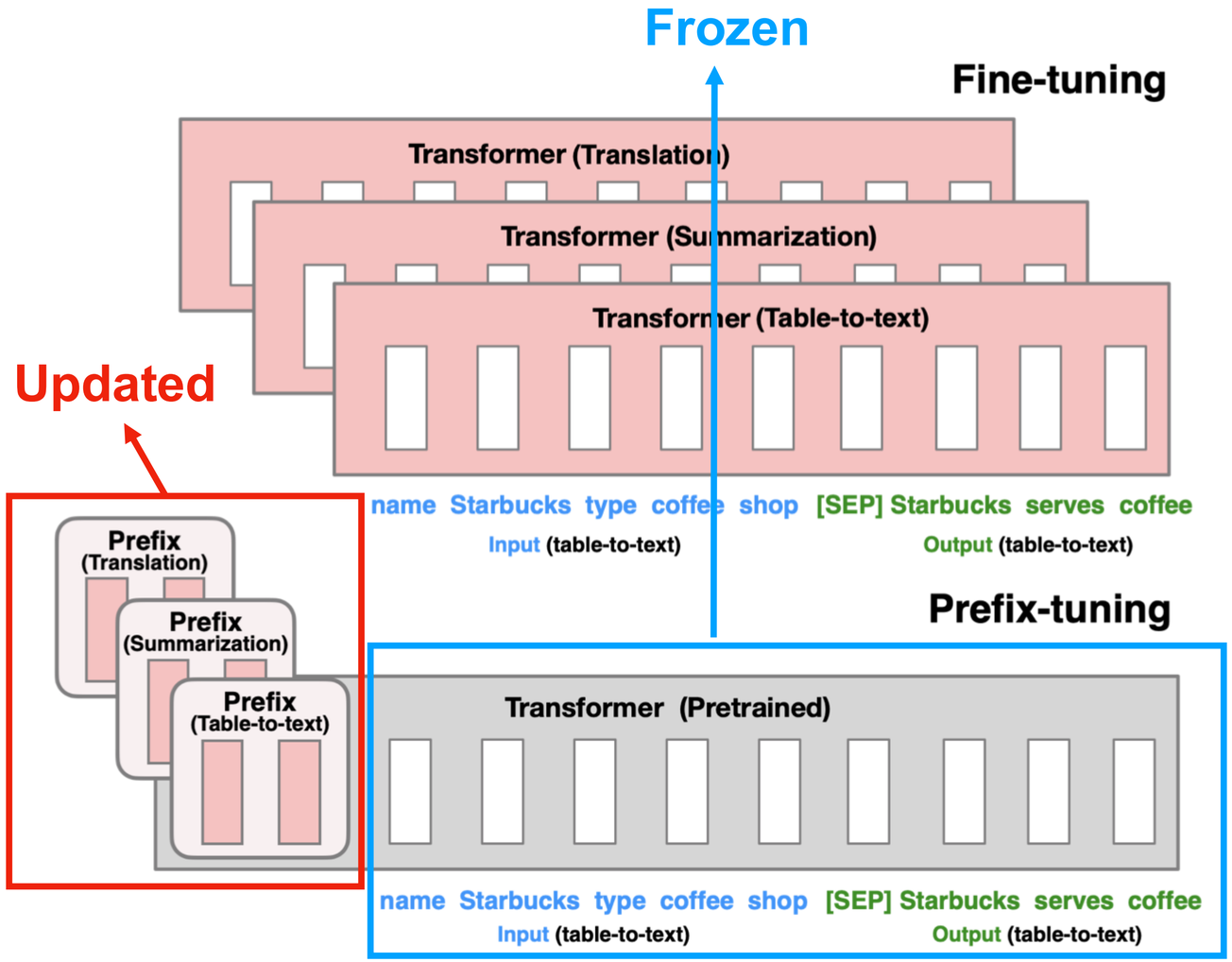
[Houlsby, 2019]



Prefix-tuning

[Li and Liang, 2021]

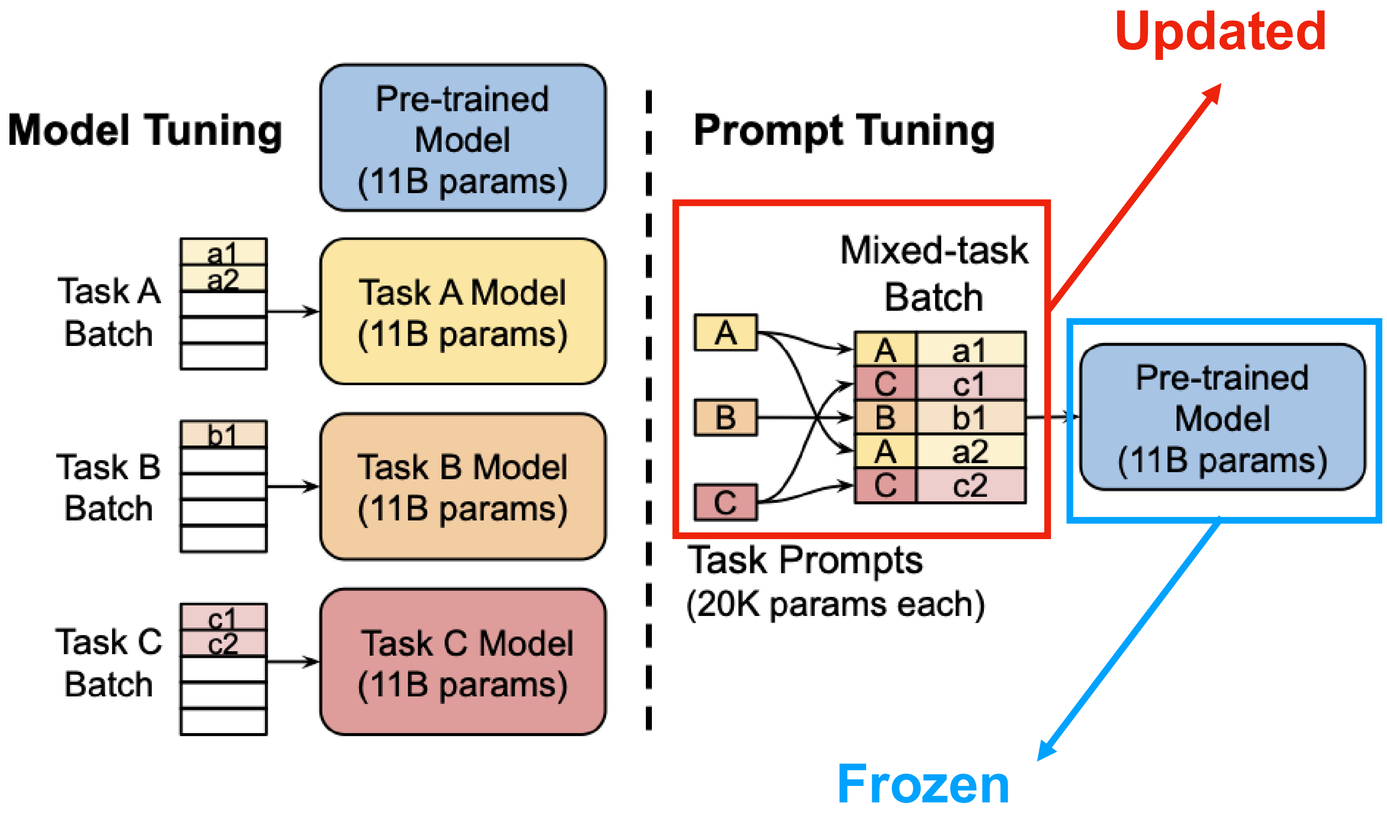
- Learning a small *continuous task-specific* vector (called the prefix) to **each transformer block**, while keeping the pre-trained LM frozen
- With 0.1% parameter is comparable to full fine-tuning, especially under low-data regime



Prompt-tuning

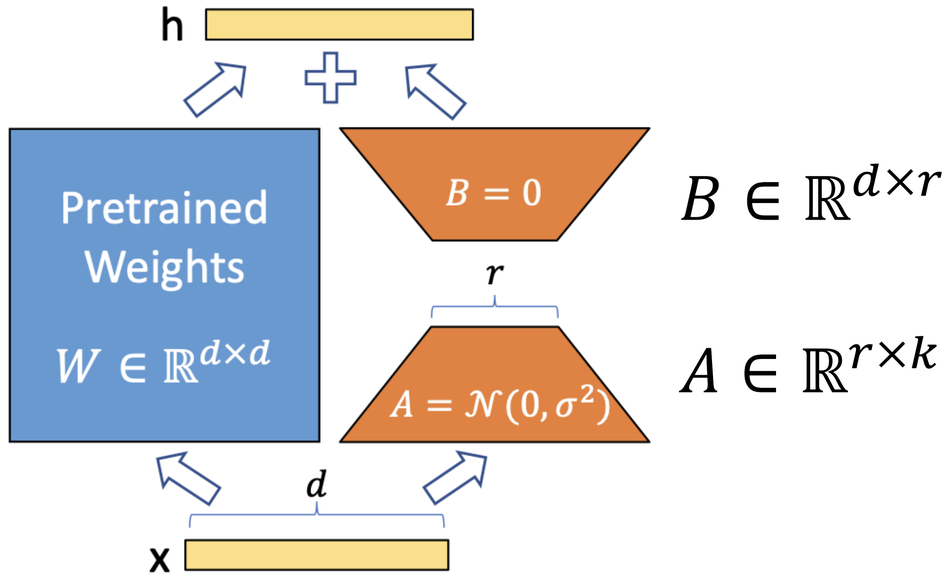
[Lester et al., 2021]

- Contemporaneous work to prefix-tuning
- A **single** "soft prompt" representation that is prepended to the **embedded input** on the encoder side
- Require **fewer** parameters than prefix-tuning

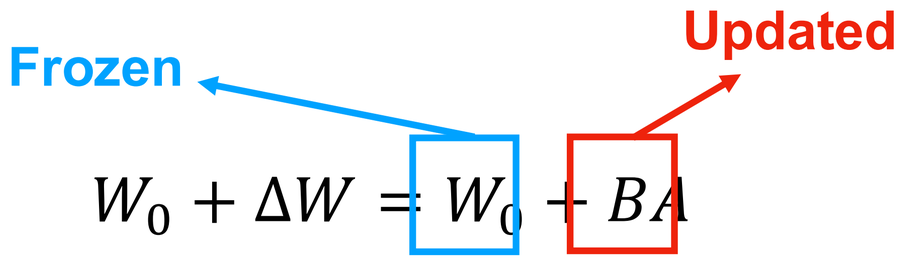


Low-Rank Adaptation (LoRA)

- **Main Idea:** learn a low-rank "diff" between the pre-trained and fine-tuned weight matrices.
- ~10,000x less fine-tuned parameters, ~3x GPU memory requirement.
- **On-par** or **better** than fine-tuning all model parameters in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3.
- **Easier** to learn than prefix-tuning.



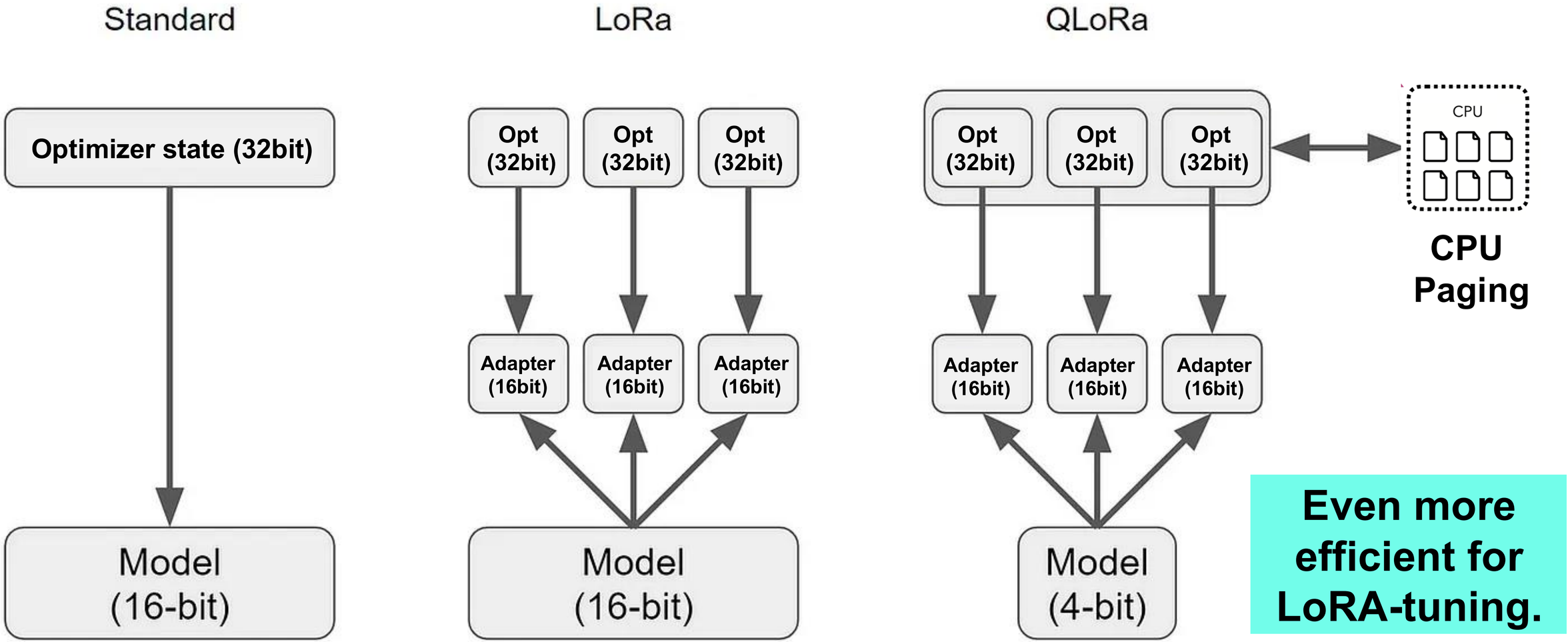
where rank $r \ll min(d, k)$



[Hu et al., 2021]

Efficient Fine-Tuning

- **Q-LoRA: Quantized LoRA**



https://arxiv.org/abs/2305.14314

Thank you!