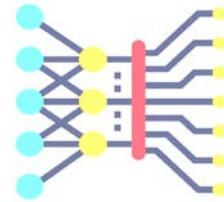


CS60010: Deep Learning

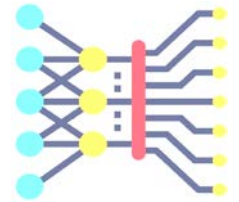
Spring 2023

Sudeshna Sarkar

Transformer- Part 3

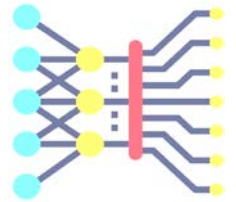
Sudeshna Sarkar

16 Mar 2023

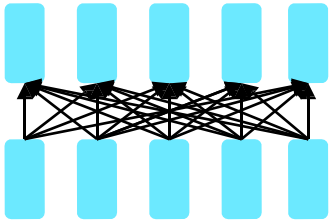


Pre-training

Pretraining for three types of architectures

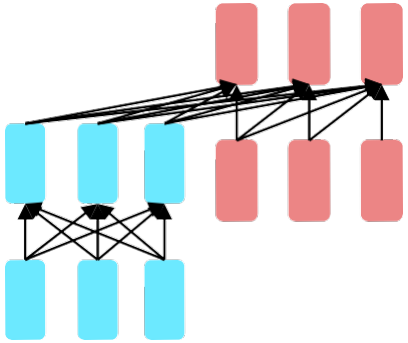


The neural architecture influences the type of pretraining, and natural use cases.



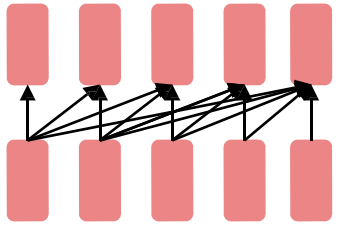
Encoders

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



Encoder-Decoders

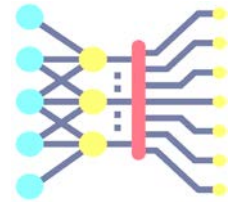
- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

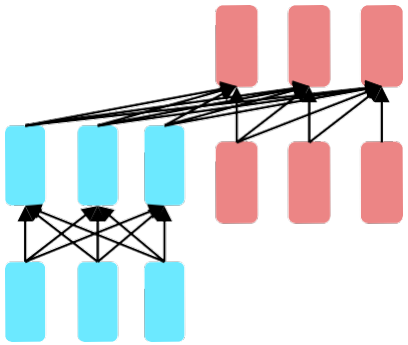
Pretraining for three types of architectures



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

Encoders

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



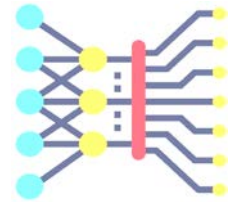
Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

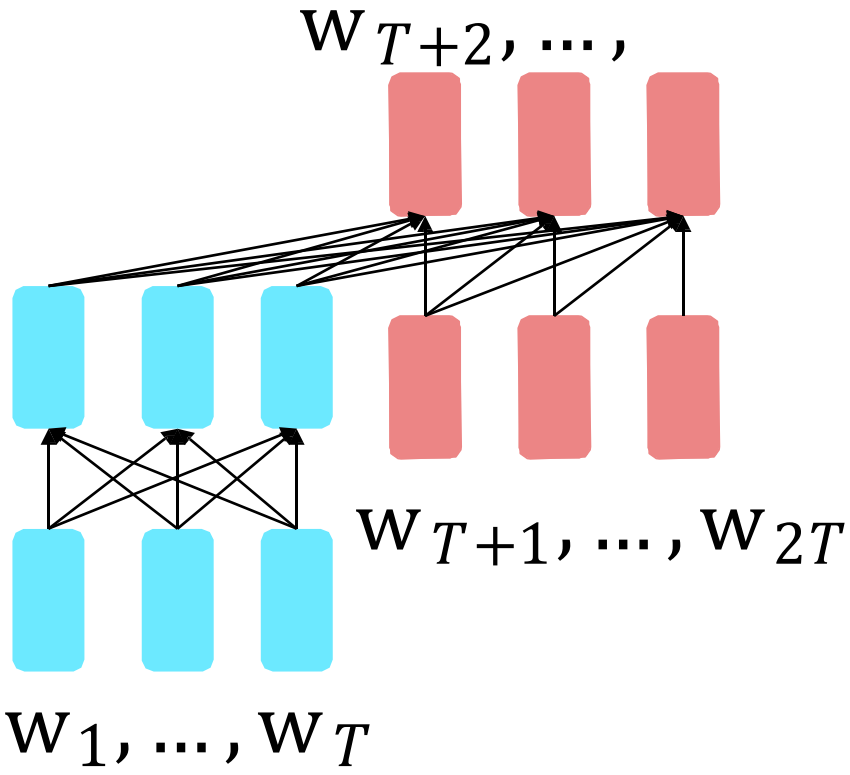
Pretraining encoder-decoders:
what pretraining objective to use?



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.

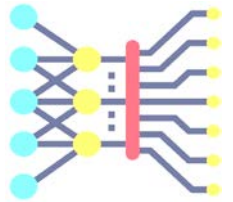
$$h_1, \ldots, h_T = \text{Encoder}\ (\text{w}_1, \ldots, \text{w}_\text{T})$$
$$h_{T+1}, \ldots, h_2 = \text{Decoder}(\ \text{w}_1, \ldots, \text{w}_T, h_1, \ldots, h_T)$$
$$y_i \sim Aw_i + b, i > T$$

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



[Raffel et al., 2018]

Pretraining encoder-decoders: what pretraining objective to use?



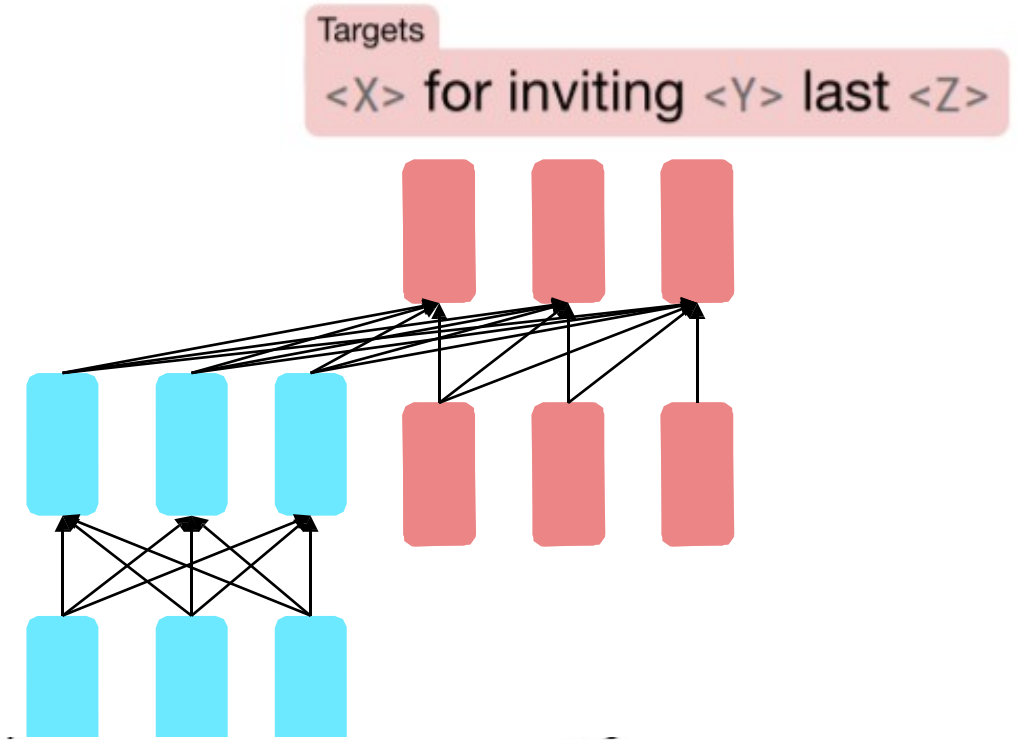
Raffel et al., 2018 proposed model: T5.
found what works best is **span corruption.**

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

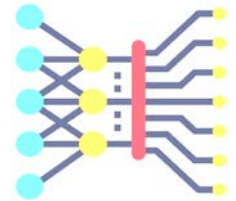
Original text

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.



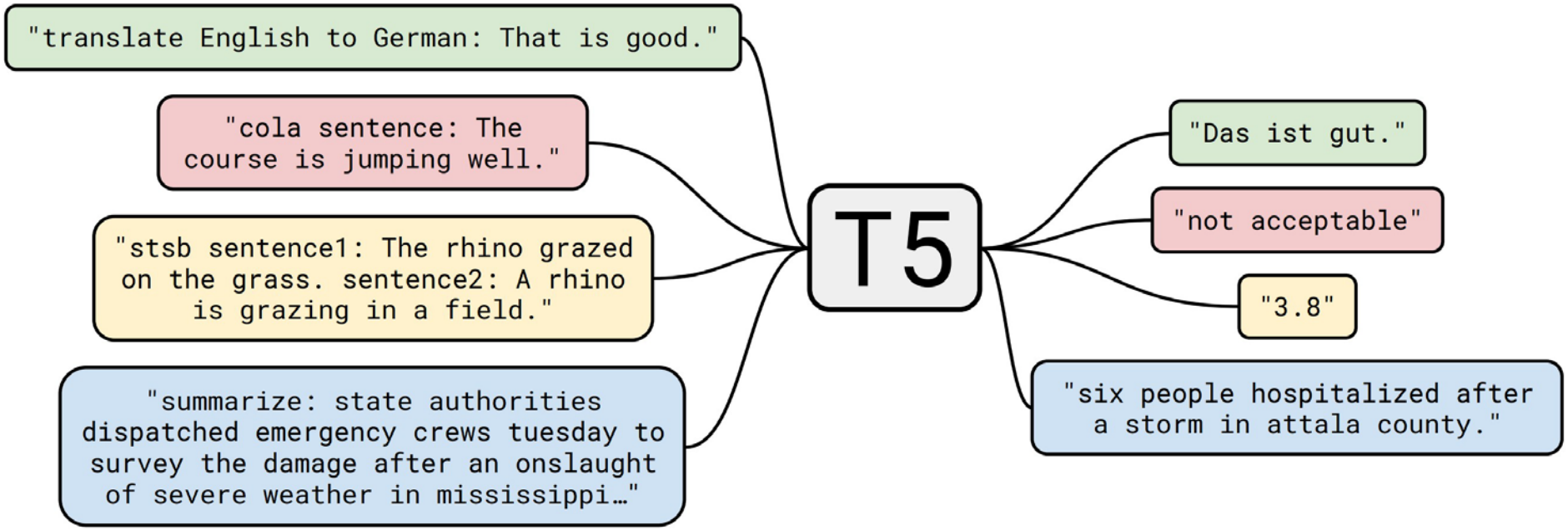
T5: “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”



- A large-scale empirical survey to determine which transfer learning techniques work best and apply these insights at scale to create a new model called the Text-To-Text Transfer Transformer (T5).
- They also introduces a new open-source pre-training dataset, called the Colossal Clean Crawled Corpus (C4).
- The T5 model, pre-trained on C4, achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of important downstream tasks.

T5

Frame many problems as sequence-to-sequence ones:



Raffel et al. (2019)

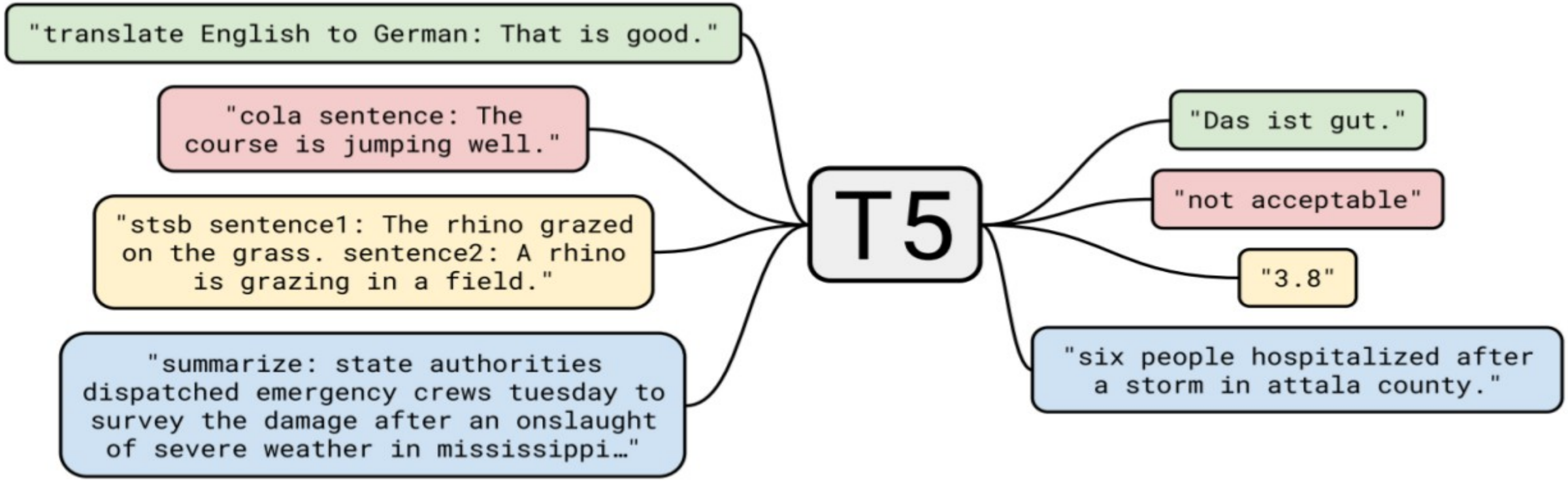


Figure 1: A diagram of our text-to-text framework. Every task we consider – including translation, question answering, and classification – is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

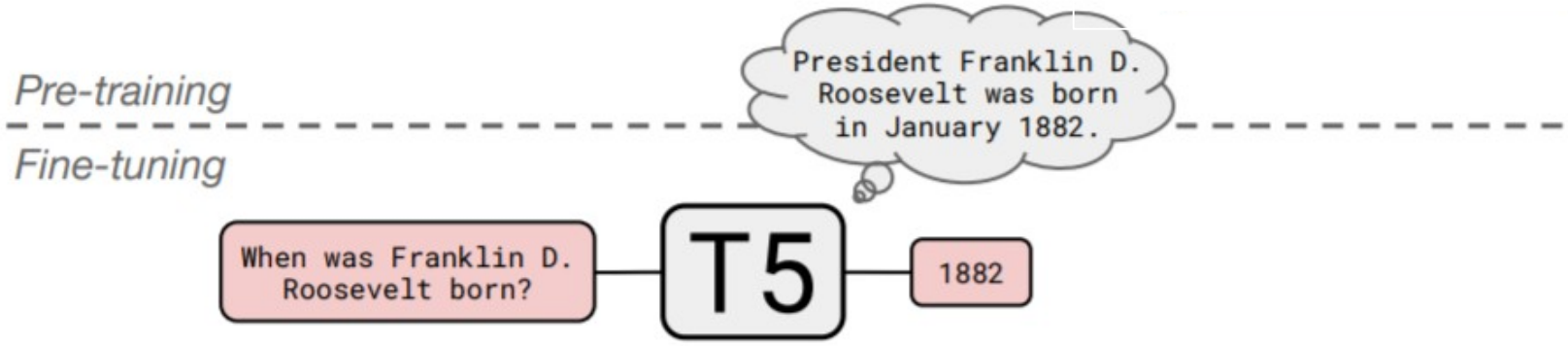
Pretraining encoder-decoders

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

NQ: Natural Questions

WQ: WebQuestions

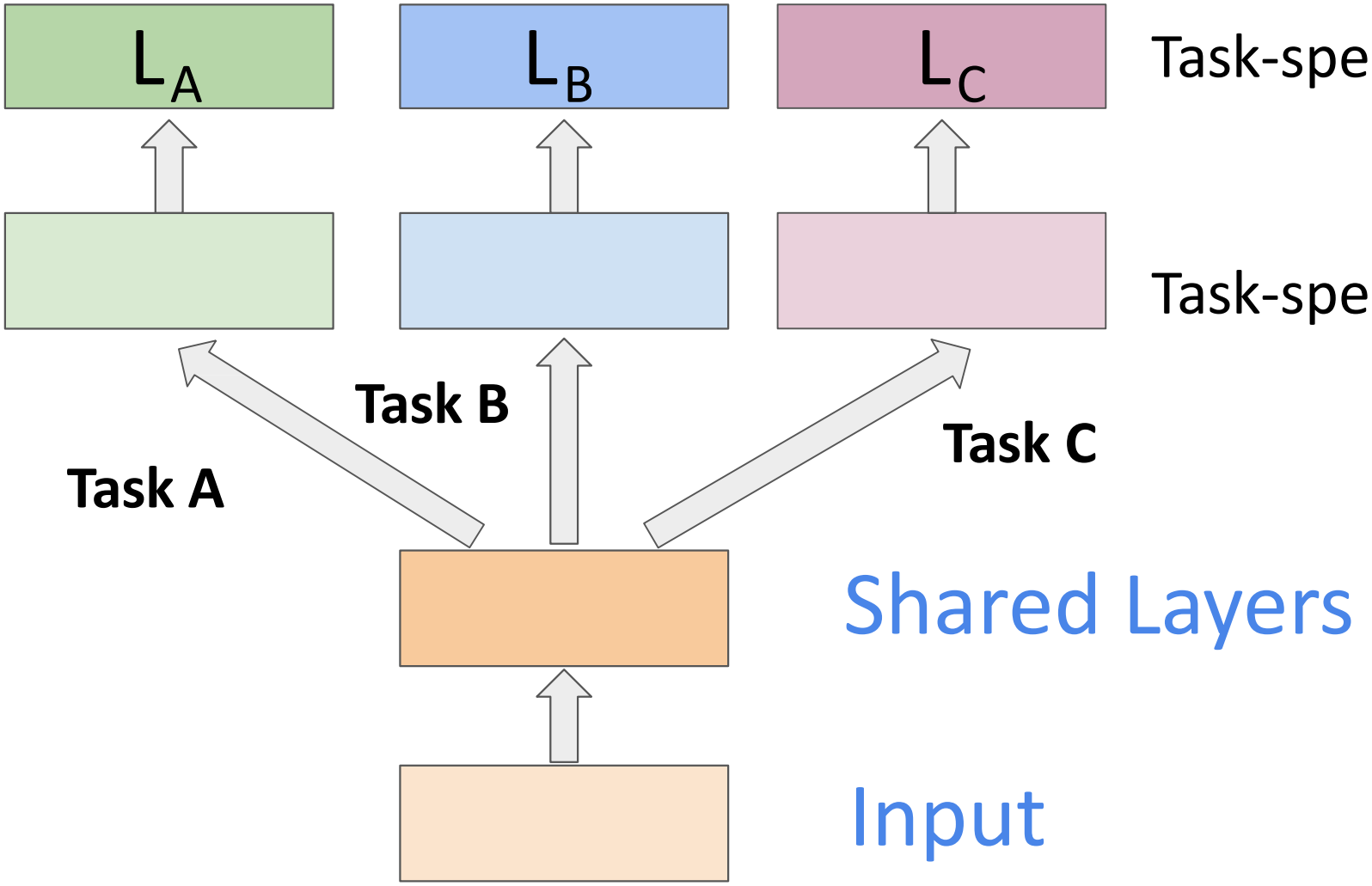
TQA: Trivia QA



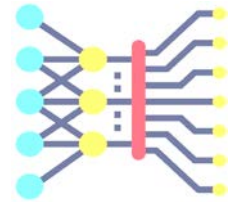
	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL +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

[Raffel et al., 2018]

Multi-task learning: Classical Paradigm

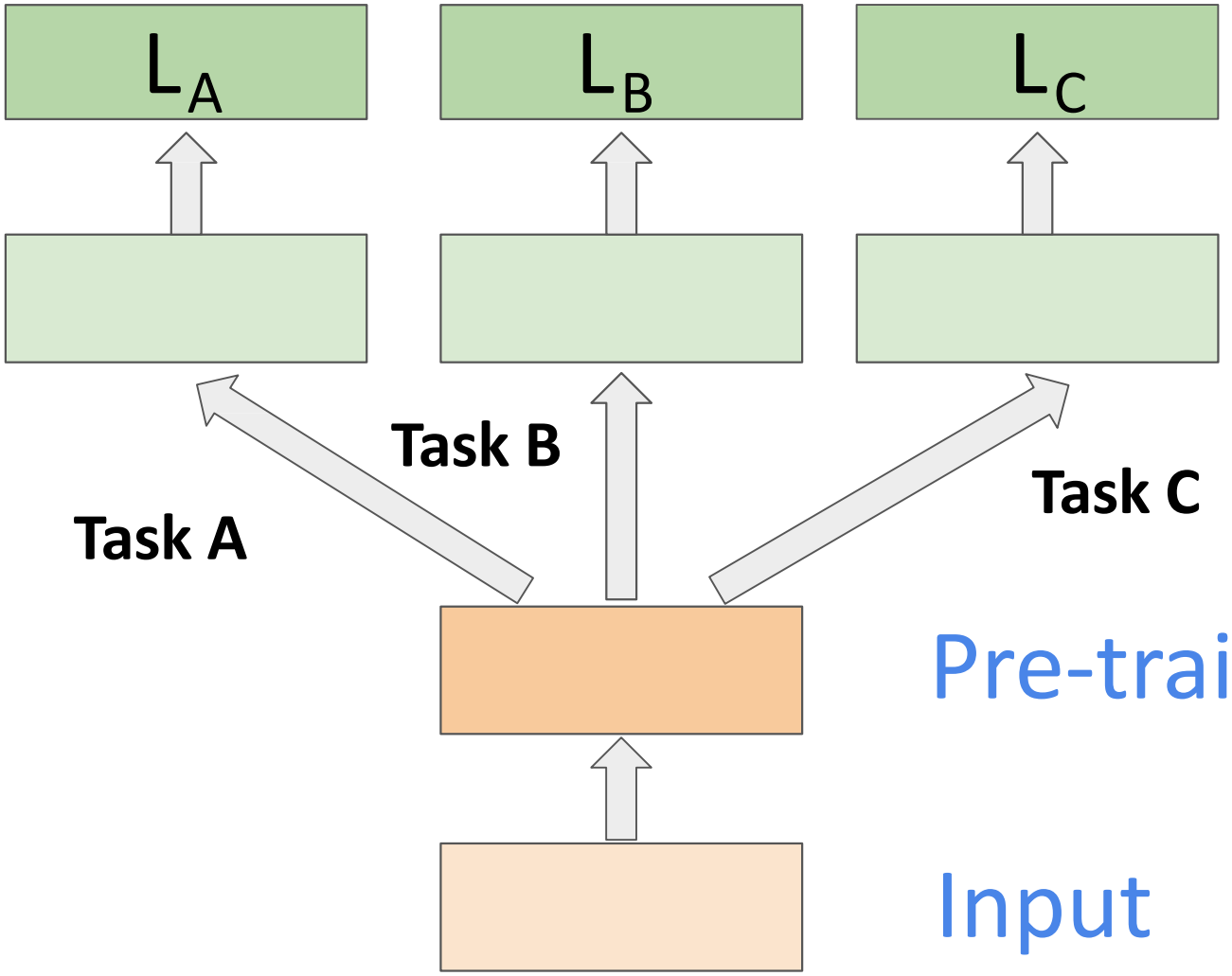


T5 (Text-to-Text Transfer Transformer): Idea



- Pre-train a Transformer Encoder-Decoder model on a large unlabeled web crawl text
- Pose every NLP task as text to text (McCann et al., 2018; Radford et al., 2019)
- Fine-tune separately for each downstream task (done in parallel)

Multi-task learning: T5 Paradigm

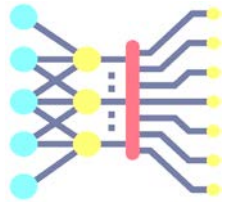


Same loss function

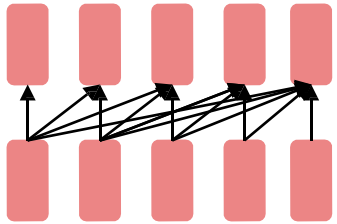
Same hyperparameters

- Cross Entropy/Max. likelihood loss for all pre-training and fine-tuning tasks
- Same hyperparameters for each task
- “Unified” vocabulary

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
- All the biggest pretrained models are Decoders.

Pretraining decoders

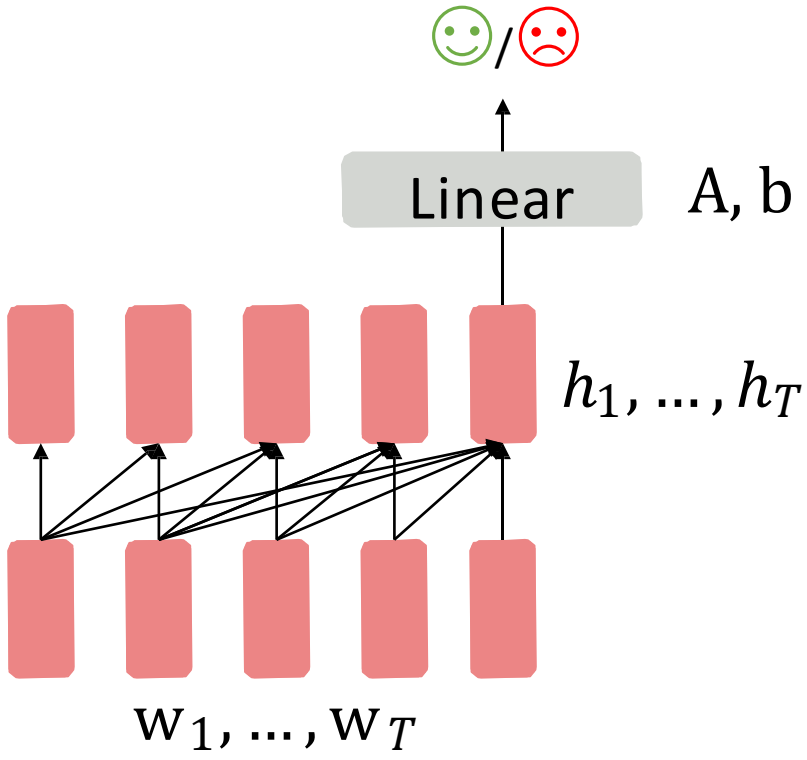
When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.

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

$$h_1, \dots, h_T = \text{Decoder}\ (\text{w}_1, \dots, \text{w}_T)$$
$$y \sim A h_T + b$$

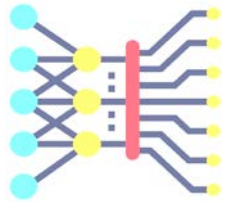
Where A and b are randomly initialized and specified by the downstream task.

Gradients backpropagate through the whole network.



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

Pretraining decoders



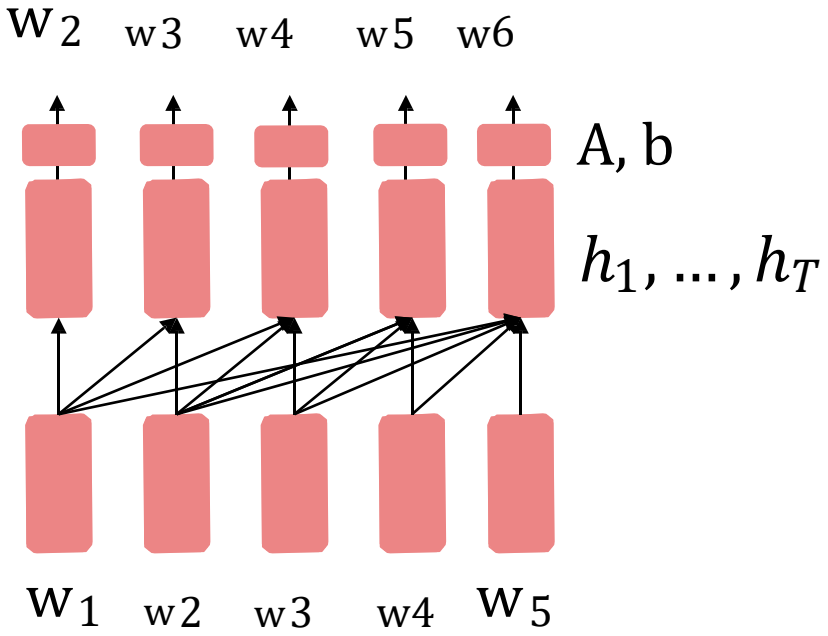
It iss natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$

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)

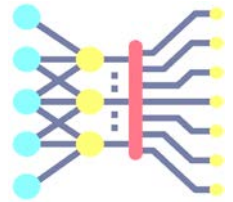
$$h_1, \ldots, h_T = \text{Decoder}\ (\text{w}_1, \ldots, \text{w}_T)$$
$$w_t \sim A h_{t-1} + b$$

where A and b are pretrained in the language model.



[Note how the linear layer has been pretrained.]

Generative Pretrained Transformer (GPT)
[Radford et al., 2018]



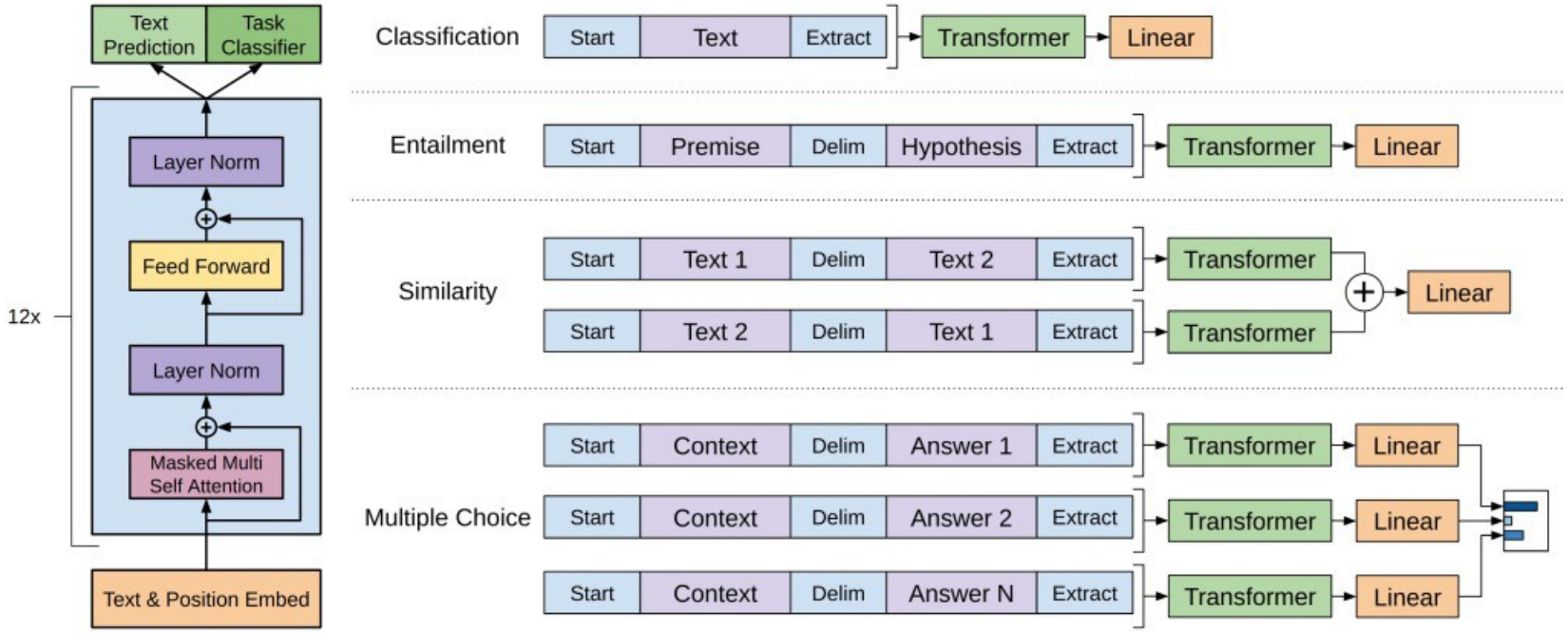
2018's GPT was a big success in pretraining a decoder!

- 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.

Generative Pretrained Transformer (GPT)

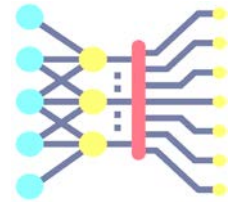
[Radford et al., 2018]

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



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

Increasingly convincing generations (GPT2)



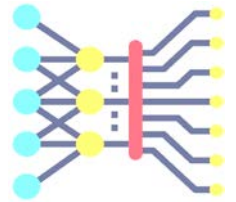
[Radford et al., 2018]

We mentioned how pretrained decoders can be used **in their capacities as language models.**

GPT-2, a larger version of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

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



So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

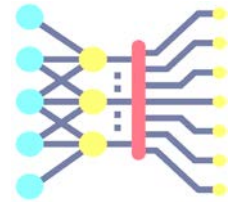
Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

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

GPT-3 has 175 billion parameters.

GPT-4 released on 15th March 2023

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

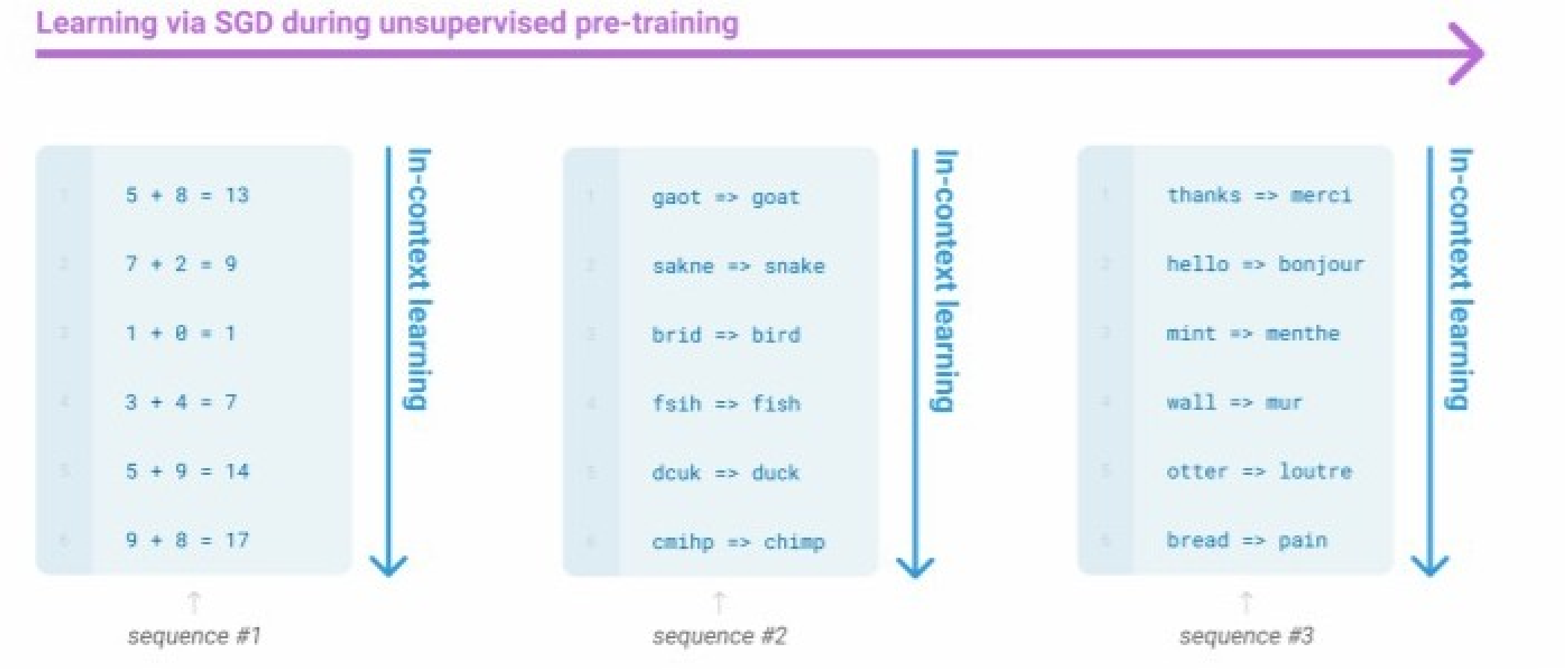
otter -> ”

Output (conditional generations):

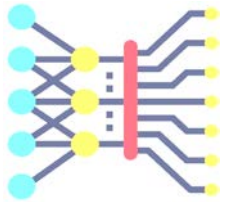
loutre…”

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 prefix as task specification and scratch pad: chain-of-thought



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

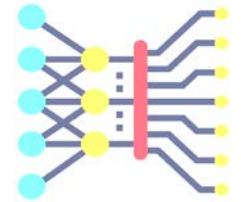
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

[Wei et al., 2023]

What kinds of things does pretraining teach?



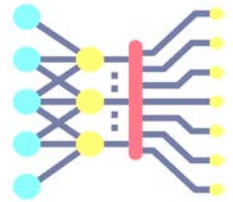
There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language. Taking our examples from the start of class:

- *Stanford University is located in _____, California.* [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.
- More on all this in the interpretability lecture!

GPT-4
released on March 14, 2023 by OpenAI

- GPT-4 is a large multimodal model (accepting image and text inputs, emitting text outputs).
- As a transformer, GPT-4 was pretrained to predict the next token (using both public data and "data licensed from third-party providers"), and was then fine-tuned with reinforcement learning from human feedback.
- Two versions of GPT-4 with context windows of 8192 and 32768 tokens

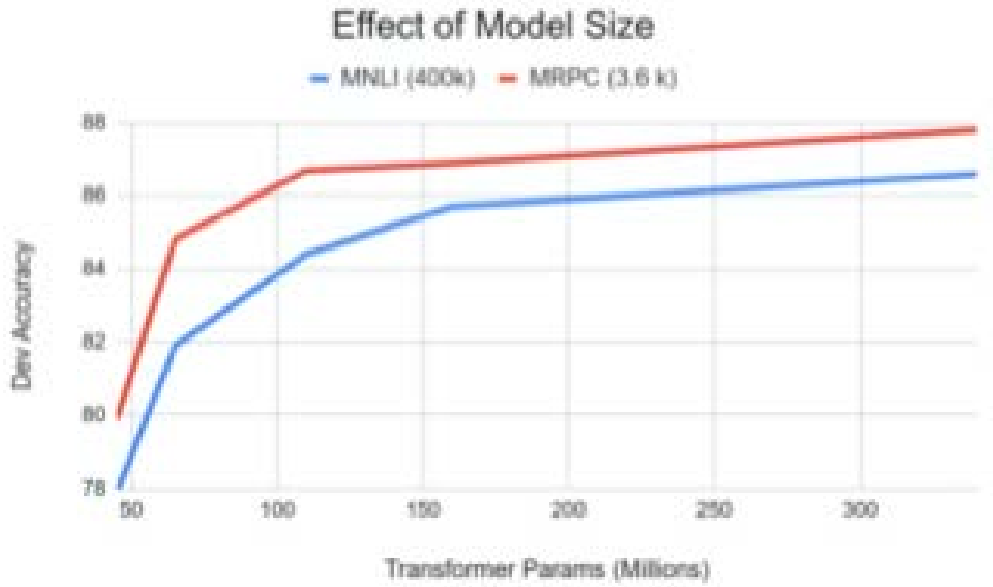
Hard to do with BERT



- BERT cannot generate text (at least not in an obvious way)

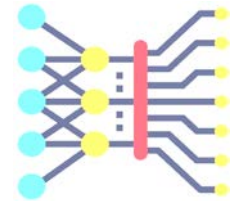
- Masked language models are intended to be used primarily for “analysis” tasks

Effects of Model Size

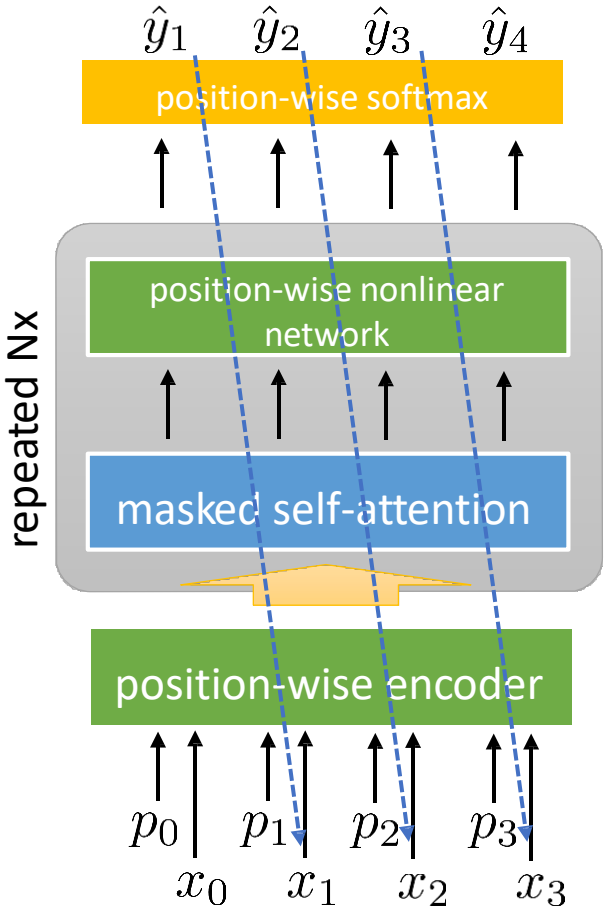


- Big models help *a lot*
- Going from 110M -> 340M params helps even on datasets with 3,600 labelled examples
- Improvements have *not* asymptoted

Slide from Jacob Delvin

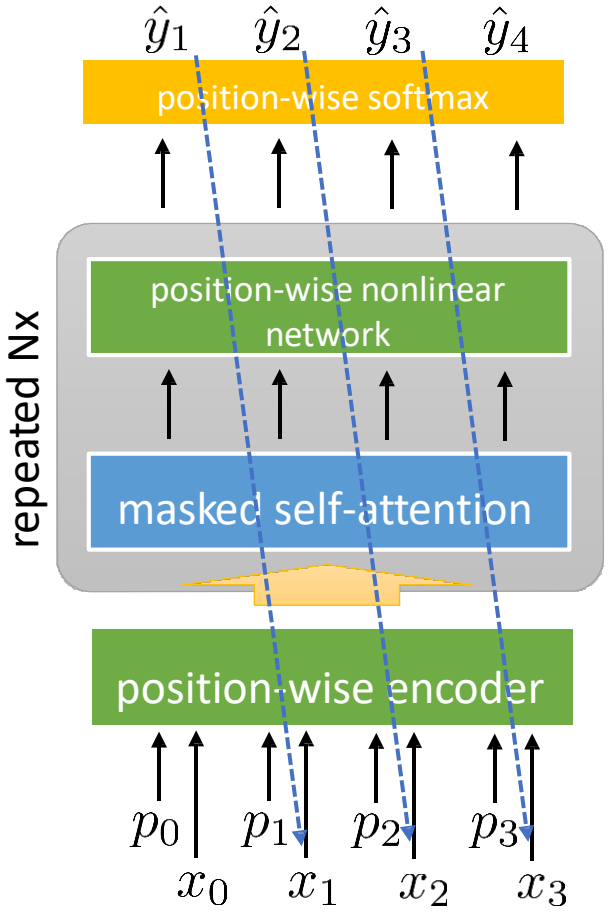


GPT et al.



- One-directional (forward) transformer models do have one **big** advantage over BERT.
- Generation is not really possible with BERT, but a forward (masked attention) model can do it!
- GPT (GPT-2, GPT-3, etc.) is a classic example of this

GPT et al.



OpenAI GPT-2 generated text

source

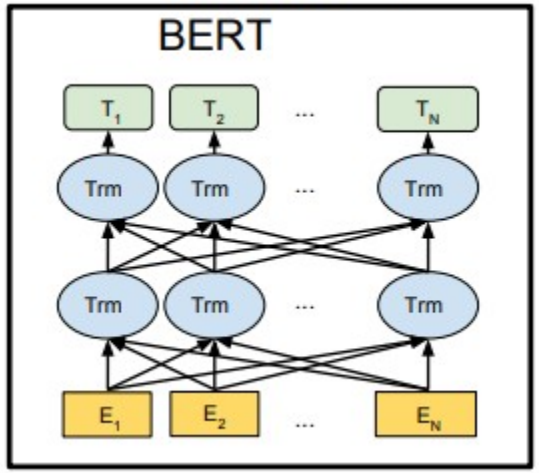
Input: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

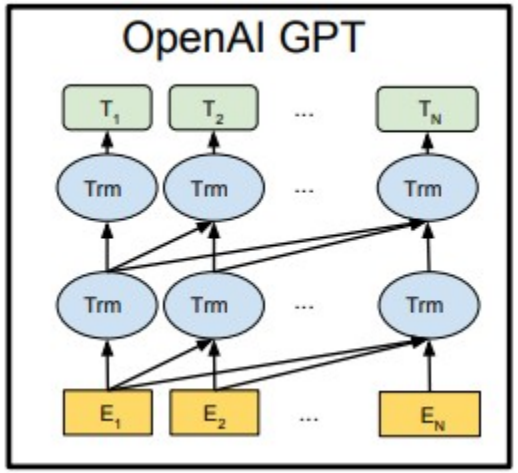
Output: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

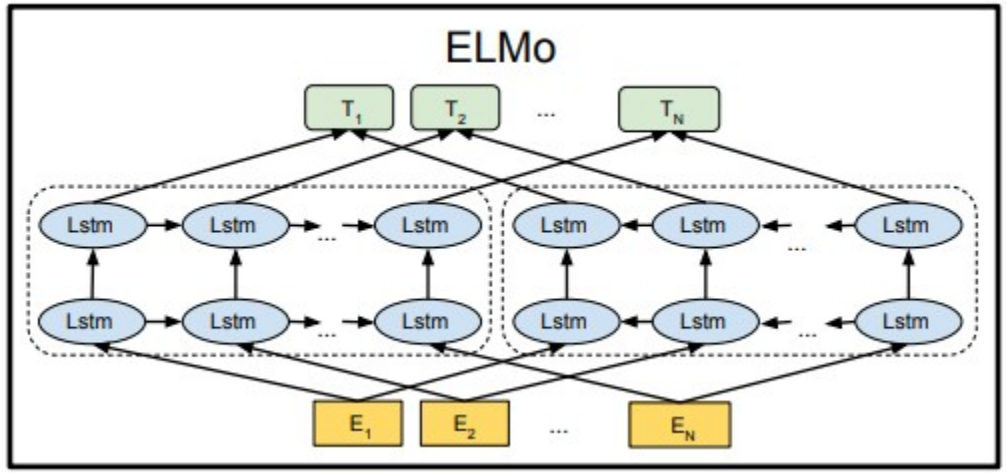
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

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.

Pretrained language models summary







bidirectional transformer

+ great representations

- can't generate text

one-directional transformer

+ can generate text

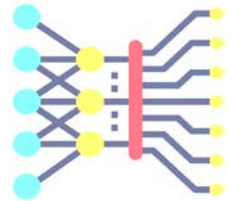
- OK representations

bidirectional LSTM

- OK representations

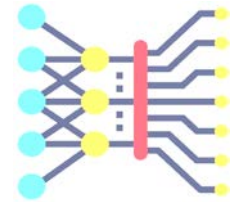
(largely supplanted by BERT)

Pretrained language models summary



- Language models can be trained on **very large and unlabeled** datasets of text.
- Internal **learned representations** depend on context: the meaning of a word is informed by the **whole sentence!**
- Can even get us representations of entire sentences (e.g., the first output token for BERT)
- Can be used to either **extract representations** to replace standard word embeddings…
- …or directly finetuned on downstream tasks (which means we modify all the weights in the whole language model, rather than just using pretrained model hidden states)

Improved variants of BERT



RoBERTa

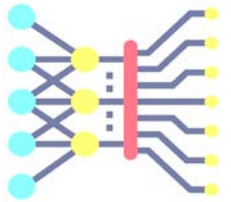
- Bigger batch size is better!
- Next sentence prediction doesn't help
- Pretrain on more data for as long as possible!

XLNet

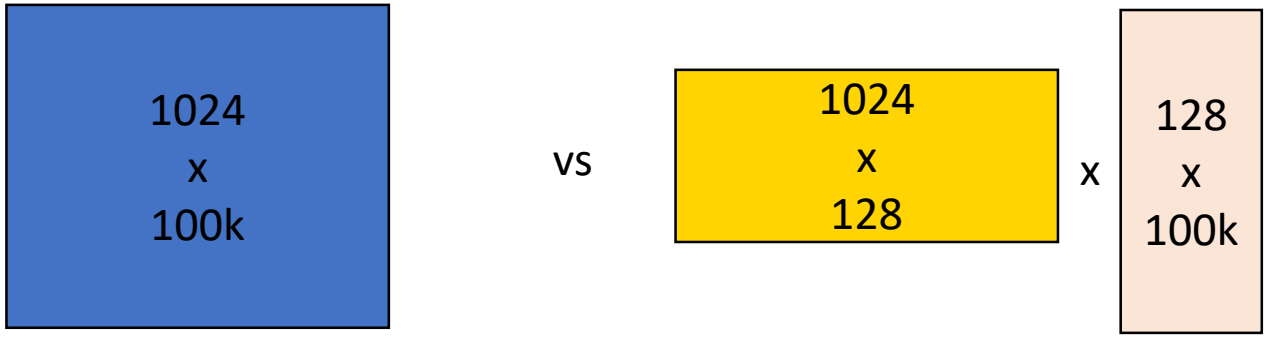
Key Ideas

- Autoregressive: use context to predict the next word
- Bidirectional context from permutation language modeling
- Self-attention mechanisms, uses Transformer-XL backbone

ALBERT

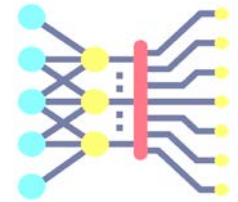


- *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations* (Lan et al, Google and TTI Chicago, 2019)
- Innovation #1: Factorized embedding parameterization
  - Use small embedding size (e.g., 128) and then project it to Transformer hidden size (e.g., 1024) with parameter matrix
- Innovation #2: Cross-layer parameter sharing
  - Share all parameters between Transformer layers



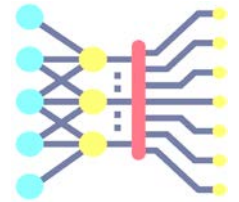
Slide from J. Delvin

GPT-2, BERT



- Transformers, GPT-2, and BERT
  1. A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
  2. But if we do not have input, we just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPT.
  3. If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.

Full Finetuning vs. Parameter-Efficient Finetuning



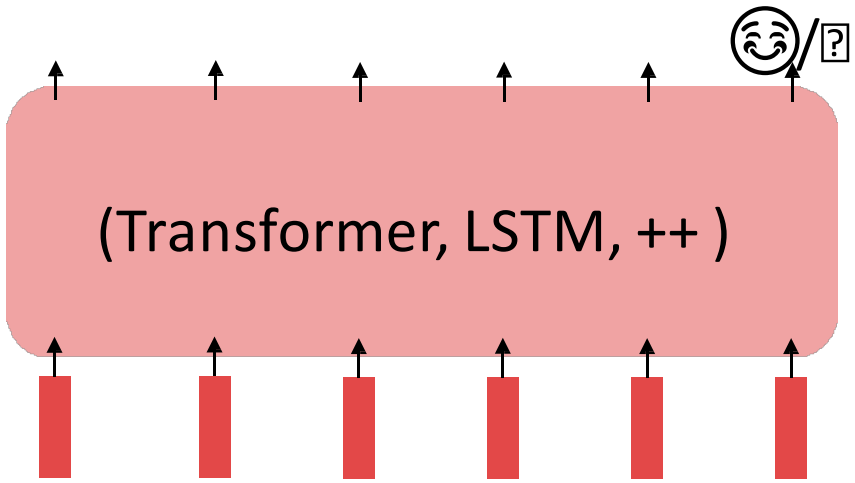
Finetuning every parameter in a pretrained model works well, but is memory-intensive.

But **lightweight** finetuning methods adapt pretrained models in a constrained way.

Leads to **less overfitting** and/or **more efficient finetuning and inference.**

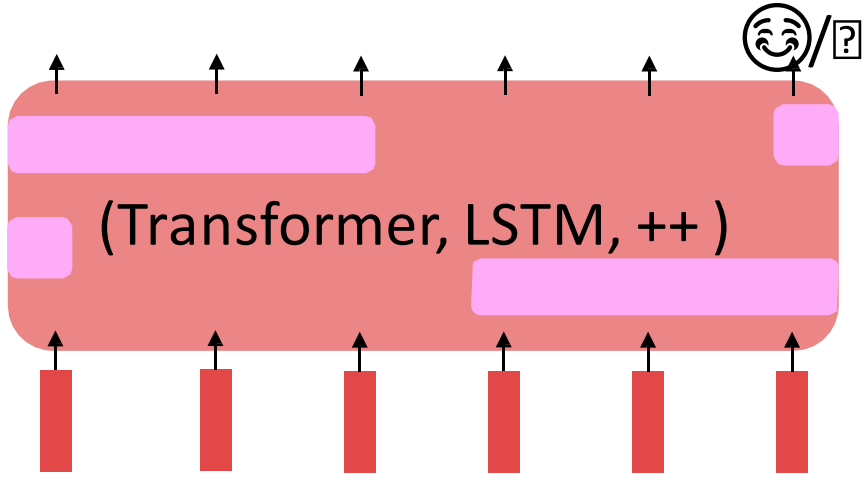
Full Finetuning

Adapt all parameters



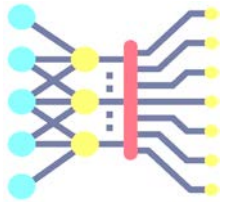
Lightweight Finetuning

Train a few existing or new parameters



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

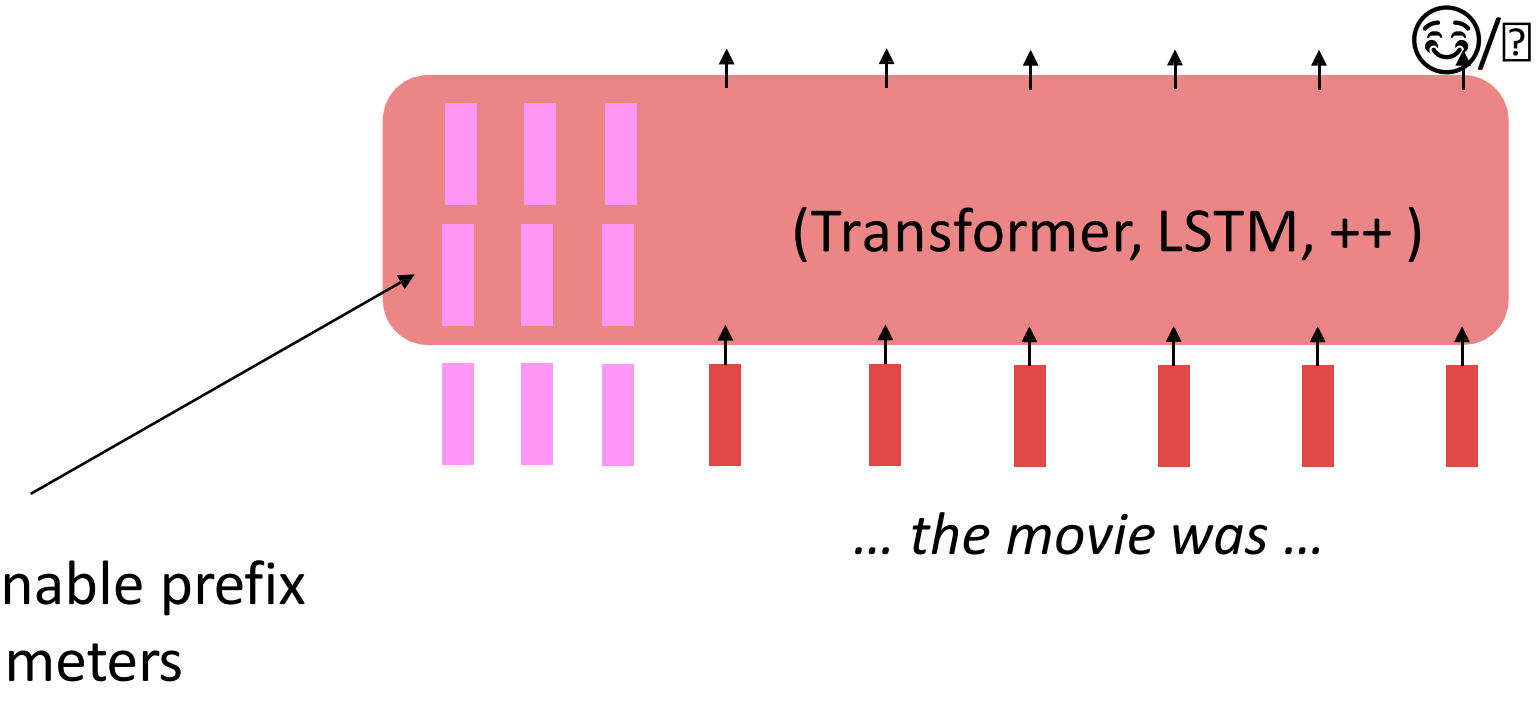
Parameter-Efficient Finetuning: Prefix-Tuning, Prompt tuning



Prefix-Tuning adds a **prefix** of parameters, and **freezes all pretrained parameters.**

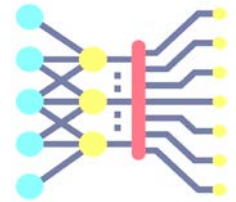
The prefix is processed by the model just like real words would be.

Advantage: each element of a batch at inference could run a different tuned model.



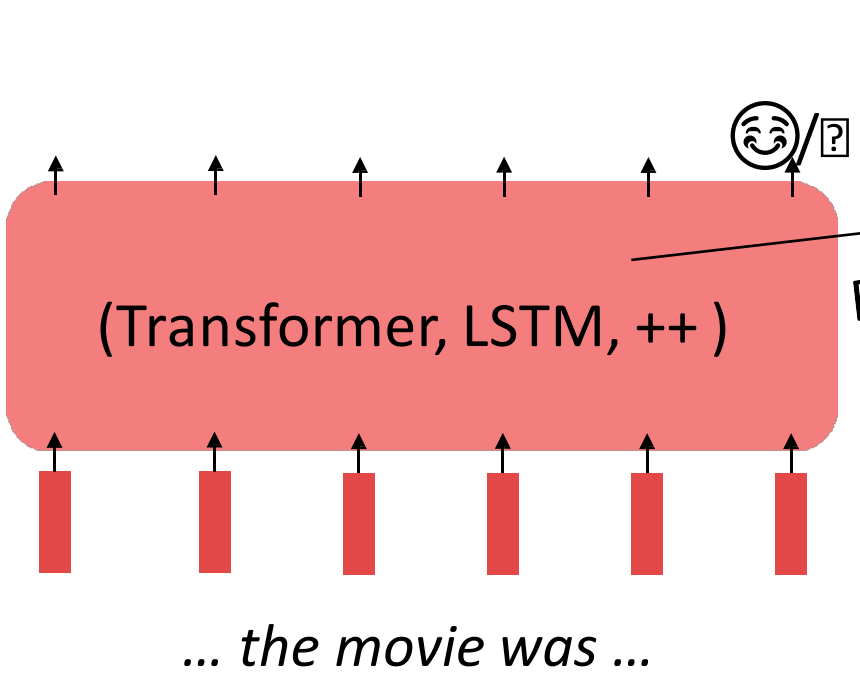
[Li and Liang, 2021; Lester et al., 2021]

Parameter-Efficient Finetuning: Low-Rank Adaptation

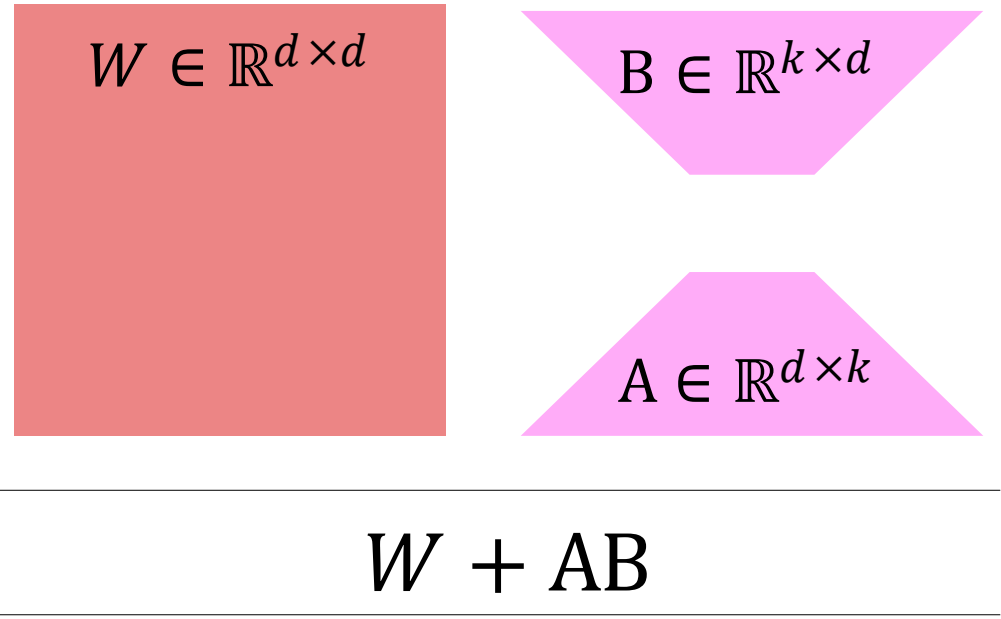


Low-Rank Adaptation Learns a low-rank "diff" between the pretrained and finetuned weight matrices.

Easier to learn than prefix-tuning.



Each weight matrix



[Hu et al., 2021]