

CSE 6331 Natural Language Processing

Transformer and Large Language Models

2026 Fall

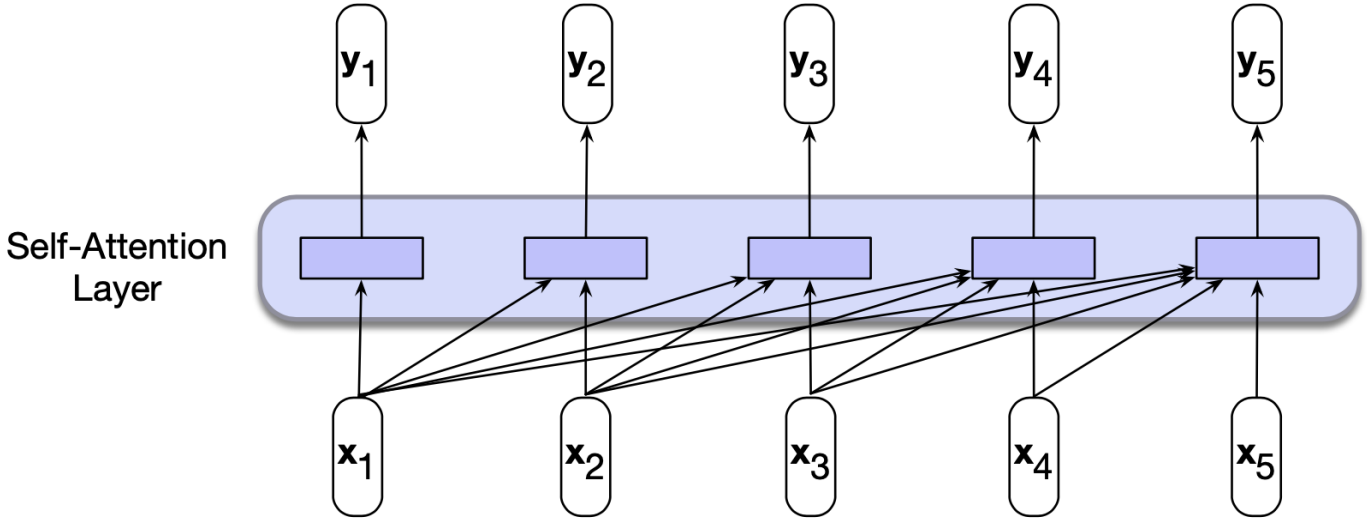
Pretraining

- Knowledge of vocabulary is acquired by "reading".
- **Distributional hypothesis**:
  - Meaning of a word can be determined by its context
  - Context can be representated by word distribution
  - Knowledge acquired can be useful in language processing *long after* its initial acquisition
- **Pretraining:**
  - the process of learning some sort of representation of meaning for words or sentences by processing very large amounts of text.
  - RNN and even Feedforward NN can be pretrained to learn language models
  - *Transformer* is a better choice

Transformer

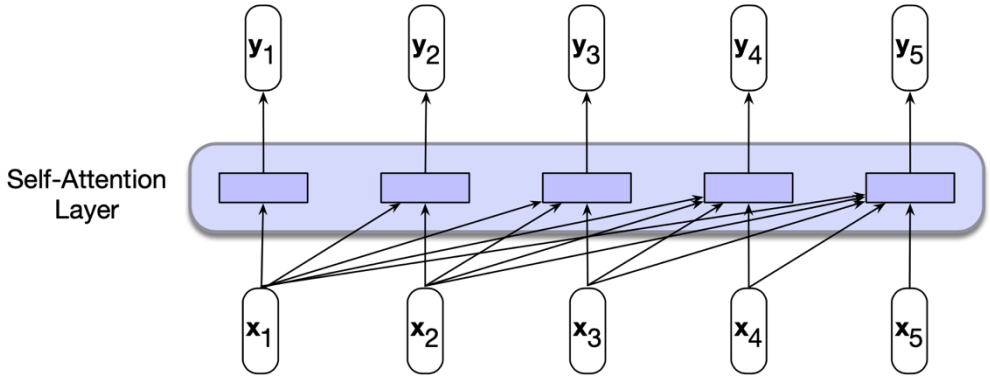
- Like LSTM, transformers can handle long-range dependencies
- But it doesn't use recurrent connections
  - recurrent architectures are hard to parallelize
  - transformers can be parallelized and are more efficient
- Two main ideas in transformers:
  - **Self-attention**
  - **Positional embeddings**

Self Attention Network



- "**Causal**" self-attention model
- Each layer maps inputs $x_1, \ldots, x_n$ to outputs $y_1, \ldots, y_n$ (equal length) → *language model* (auto-regressive generation)
- y_i depends on $x_1, \ldots, x_i$, and can be computed independently from other y_j.→ *parallelism*

A SIMPLE SELF-ATTENTION



- The computation of $\boldsymbol{y}_i$ depends on the comparison tween $\boldsymbol{x}_i$ with $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, up to $\boldsymbol{x}_i$ itself.
- Simple form: $score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$ (dot product)
- The larger the score, the the more similar they are.
- Attention weight vector:

$$\begin{aligned} \alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i \\ &= \frac{\exp(\text{score}(\mathbf{x}_i, \mathbf{x}_j))}{\sum_{k=1}^{i} \exp(\text{score}(\mathbf{x}_i, \mathbf{x}_k))} \;\; \forall j \leq i \end{aligned}$$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{x}_j$$

The self-attention in transformers

- Each input $\boldsymbol{x}_i$ plays three roles:
- Query: *current focus of attention* when compared to all preceding input;
- Key: *preceding input* being compared to current focus of attention;
- Value: used to compute the output of current focus of attention;

$$\mathbf{q}_i = \mathbf{W}^{\mathbf{Q}}\mathbf{x}_i;\ \ \mathbf{k}_i = \mathbf{W}^{\mathbf{K}}\mathbf{x}_i;\ \ \mathbf{v}_i = \mathbf{W}^{\mathbf{V}}\mathbf{x}_i$$

- $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are vectors of d-dimension.
- $W^Q \in \mathbb{R}^{d\times d}, W^K \in \mathbb{R}^{d\times d}, W^V \in \mathbb{R}^{d\times d}$

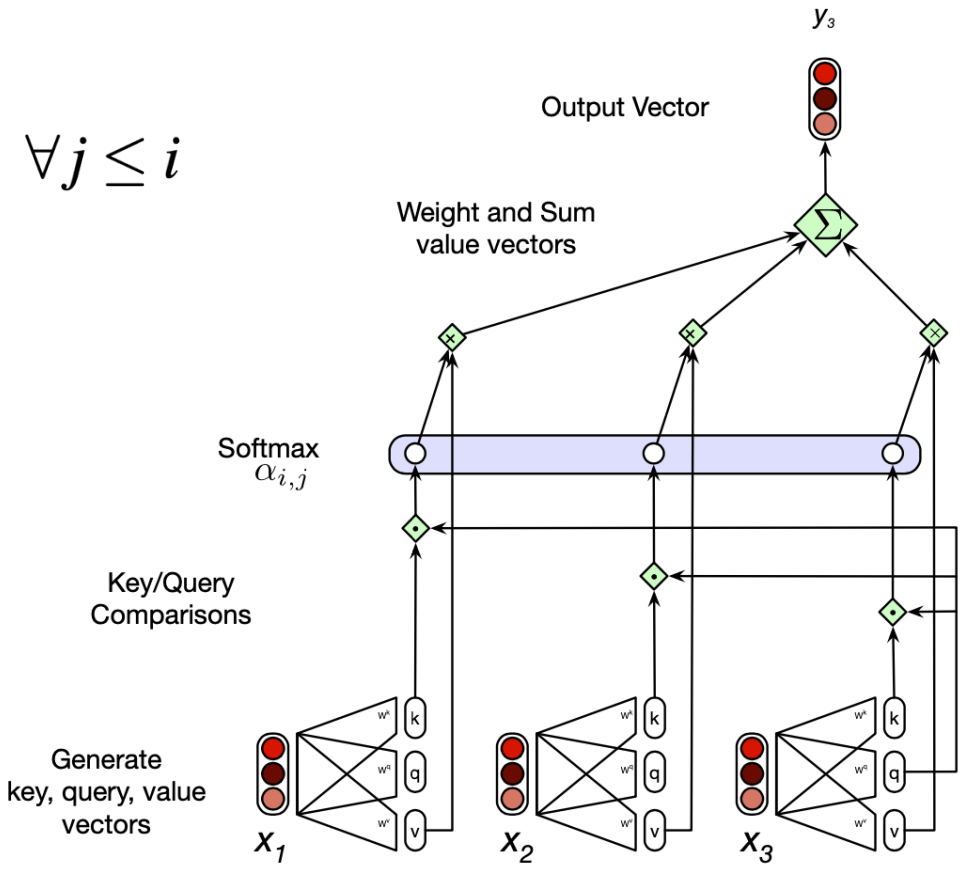
Reminds us of the target and context roles of word embedding?

The self-attention in transformers

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{q}_i \cdot \mathbf{k}_j$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \ \forall j \leq i$$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$



Scale the score down:

$$score(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d}}$$

Computing all inputs together:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}}; \ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}}; \ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

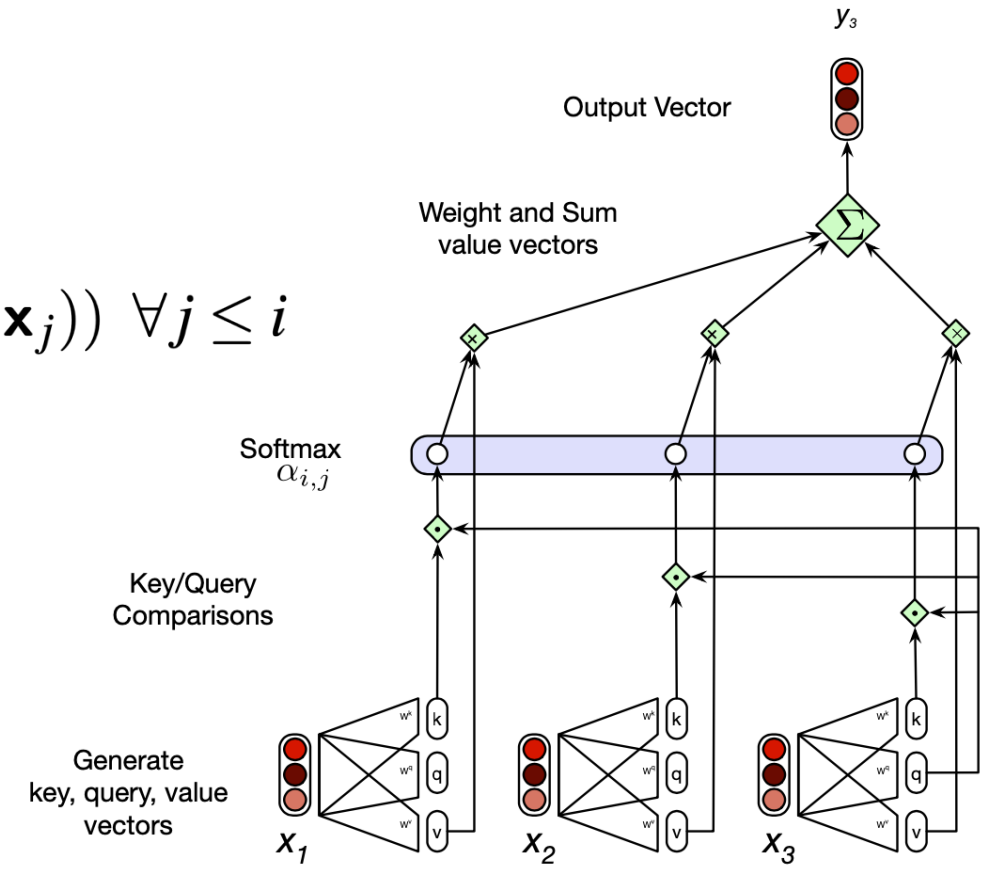
Quiz: Complexity of Attention (3 pts)

- What is the time complexity of computing attention for all input, in terms of the length of input N?

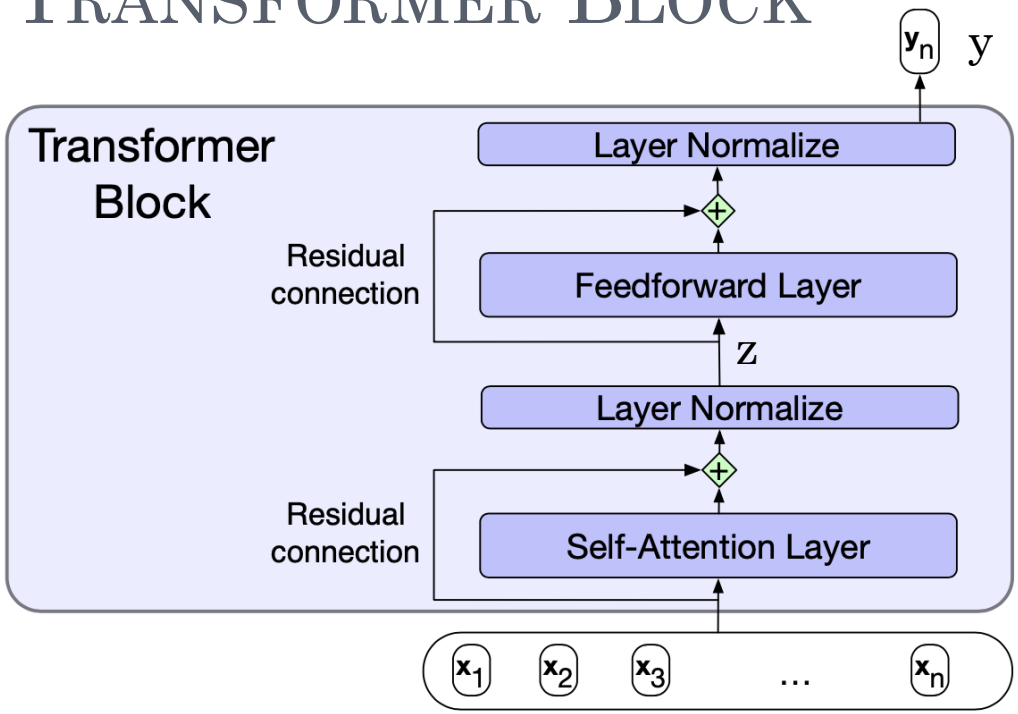
$$score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i$$

$$\mathbf{y}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j$$



Transformer Block



$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{SelfAttention}(\mathbf{x}))$$
$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFN}(\mathbf{z}))$$

- Residual connection by-passes the information from lower layer to higher layer without going through the intermediate layer
- Residual info is summed with the output the intermediate layer.

Layer Norm

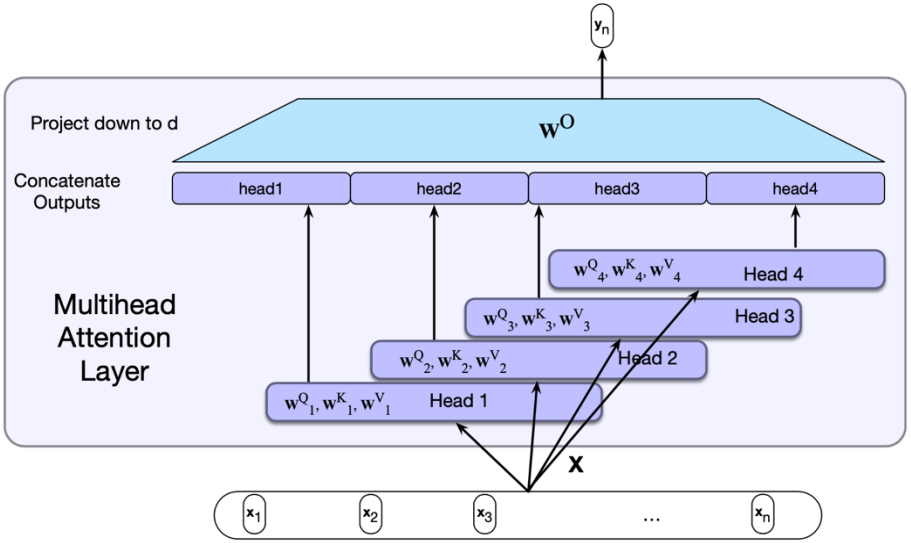
- Layer normalization can be any normalization that keeps the values of hidden layers in a range that is "gradient friendly."
- Mean: $\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$ (d_h is the width of input/output)
- Standard dev: $\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$
- normalized: $\hat{\mathbf{x}} = \frac{(\mathbf{x} - \mu)}{\sigma}$

$$\text{LayerNorm} = \gamma\,\hat{\mathbf{x}} + \beta$$

trainable params

Multi-head Attention

- Different words in a sentence can related to each other in different ways:
  - syntactic
  - semantic
  - discourse
  - …



- Instead of one self-attention layer, multiple self-attention layers (called **heads**) in parallel (concatenated):

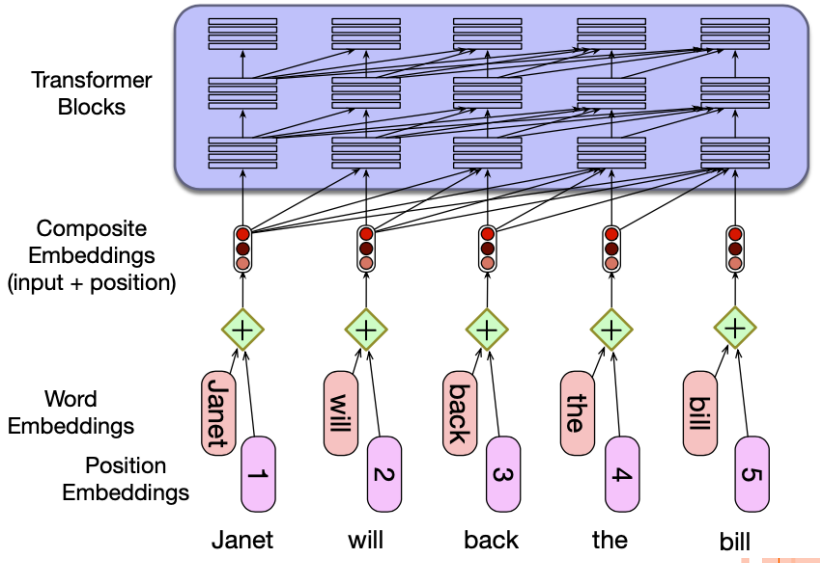
$$\text{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_i^Q \; ; \; \mathbf{K} = \mathbf{X}\mathbf{W}_i^K \; ; \; \mathbf{V} = \mathbf{X}\mathbf{W}_i^V$$

$$\mathbf{head}_i = \text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$$

Positional Embedding

- In a transformer, input tokens are in parallel.
- There's no notion of order at all!
- Idea: add a positional embedding to word embedding to get the new input embedding
- Positional embeddings are learned just like word embedding:
  - randomly initialized
  - one vector for each position such as 1, 2, 3
  - Problem: far positions such as 100, 200 are poorly trained



Quiz: Multi-head vs Positional Embedding (3 pts)

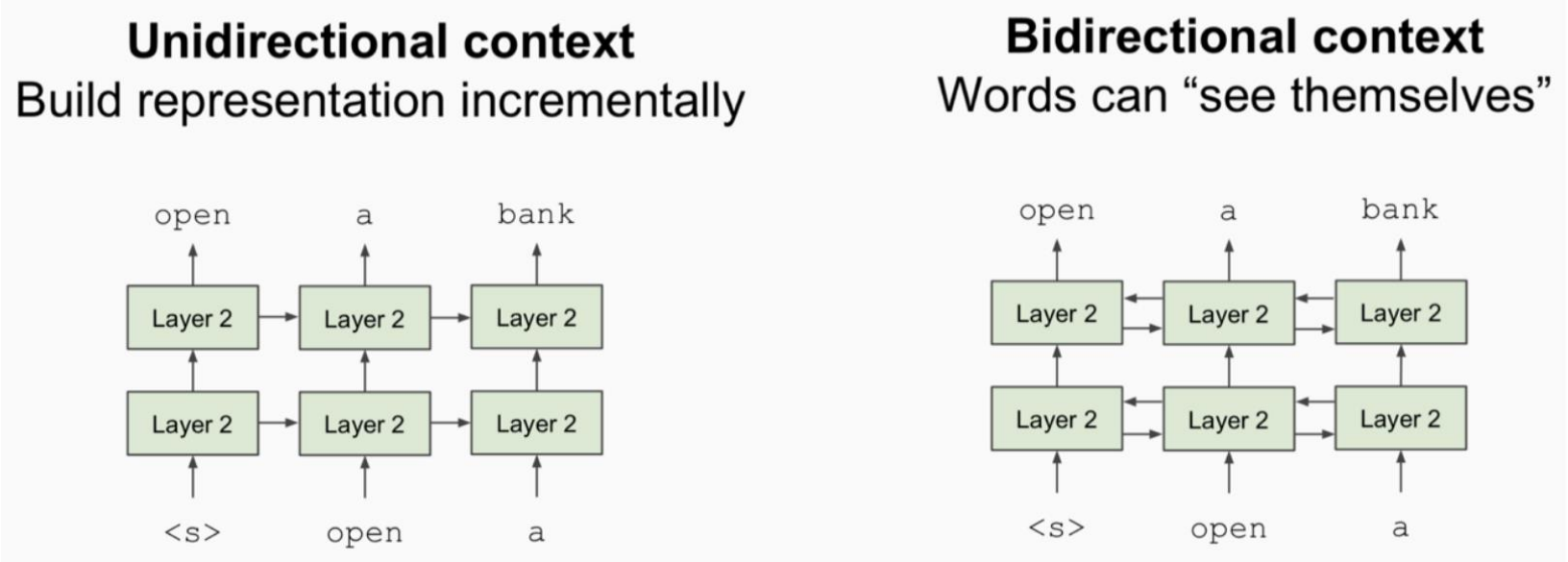
- Why do we concatenate the multiple head vectors together to get the overall attention, but simply add (element-wise) the word vector and positional vector to create the new input vector?

BERT

- "Bidirectional Encoder Representations from Transformers"
- First released in Oct 2018.
- NAACL'19: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- BERT provides contextualized word embedding
- An improvement from ELMo:
  - bidirectional context vs unidirectional context
  - Transformers vs LSTMs
  - The weights are not frozen, called *fine-tuning*

Bidirectional Encoders

- Language models only use left context or right context (although ELMo used two independent LMs from each direction).
- Language understanding is bidirectional



Masked Language Models

- How to pretrain the language model?
- Solution: Mask out 15% of the input words, and then predict the masked words



- Too little masking: too expensive to train
- Too much masking: not enough context

Quiz: Masking (2 pts)

- Why is it too expensive to train if we mask too little in the input text?

Masked Language Models

- Because BERT will never see [MASK] in real-world data, training data is a little more complicated:

  - Rather than *always* replacing the chosen words with `[MASK]`, the data generator will do the following:
  - 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
  - 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
  - 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Next Sentence Prediction (NSP)

- Always sample two sentences, predict whether the second sentence follows the first one.

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

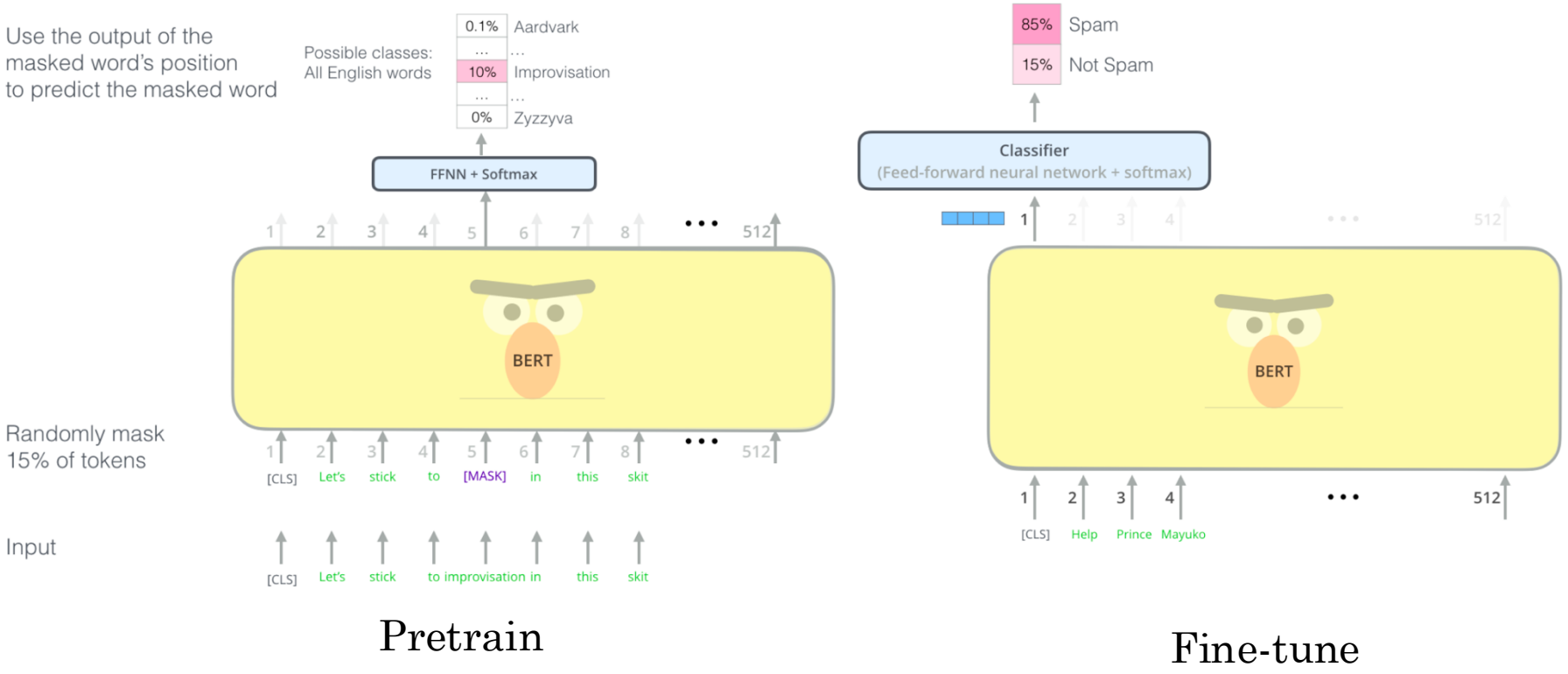
Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

- Recent paper shows that NSP is not necessary…

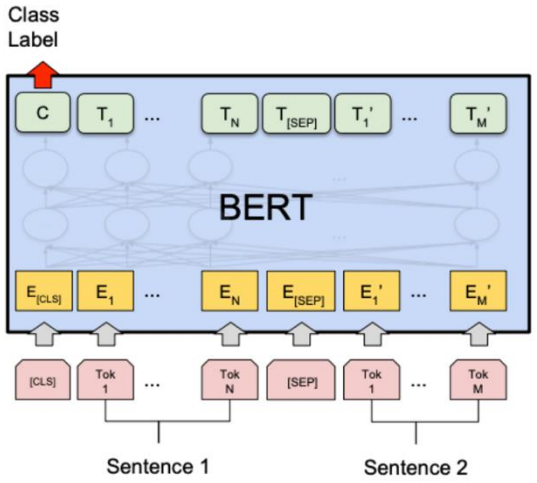
(Joshi*, Chen* et al, 2019) :SpanBERT: Improving Pre-training by Representing and Predicting Spans
(Liu et al, 2019): RoBERTa: A Robustly Optimized BERT Pretraining Approach

Pretraining and Fine-tuning

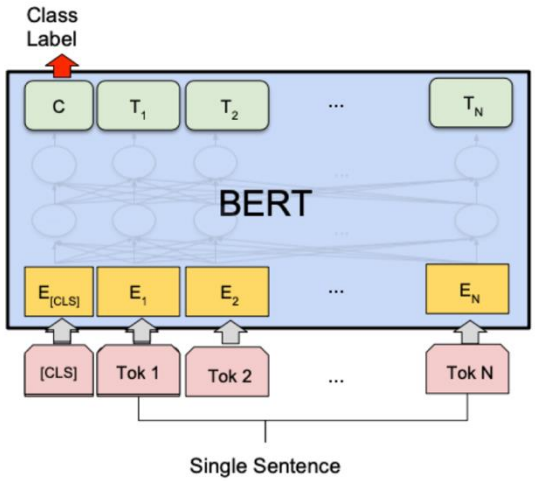
- Key idea: all the weights are fine-tuned on downstream tasks



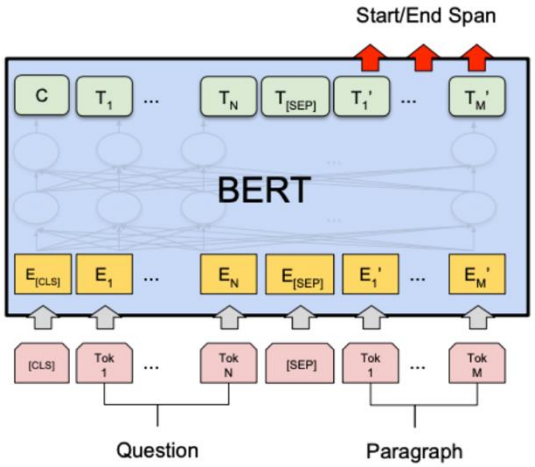
BERT Applications



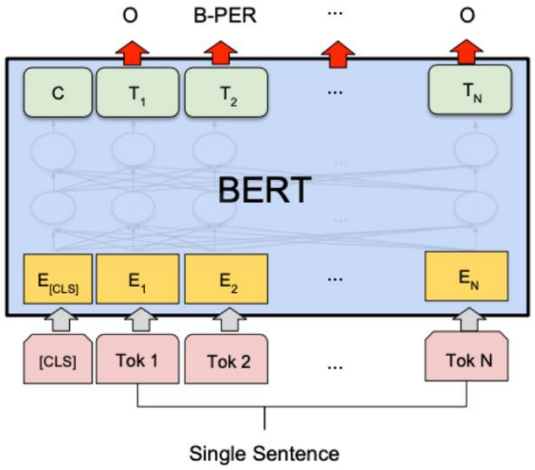
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



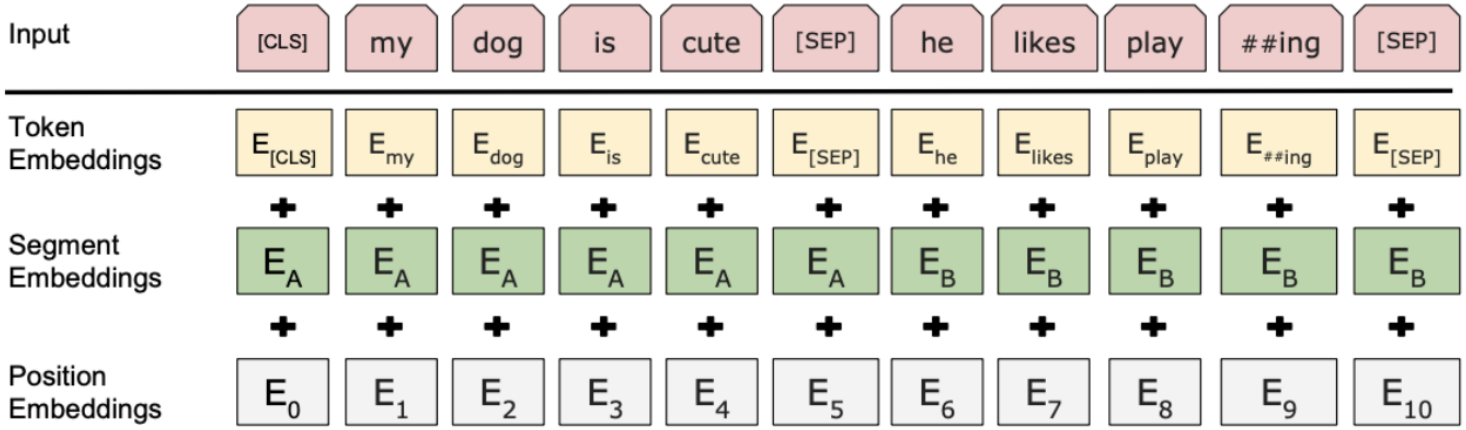
(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT More Details

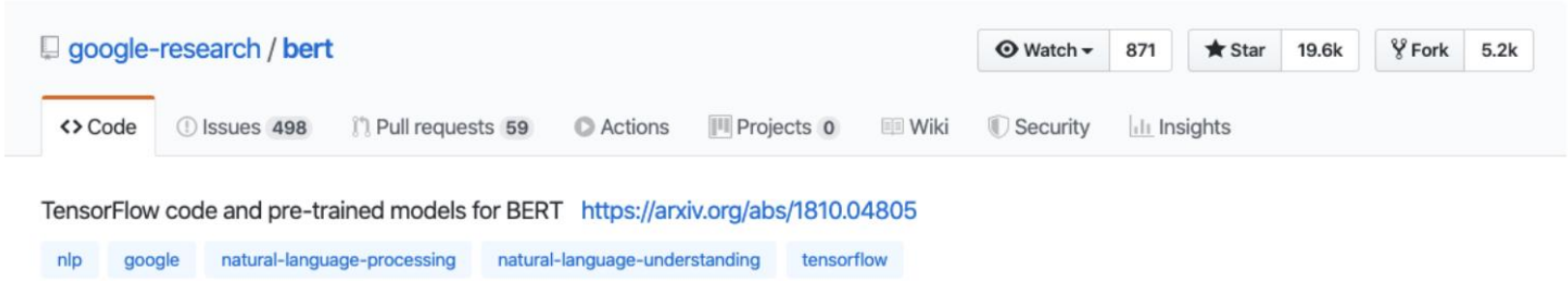
- Input representations:



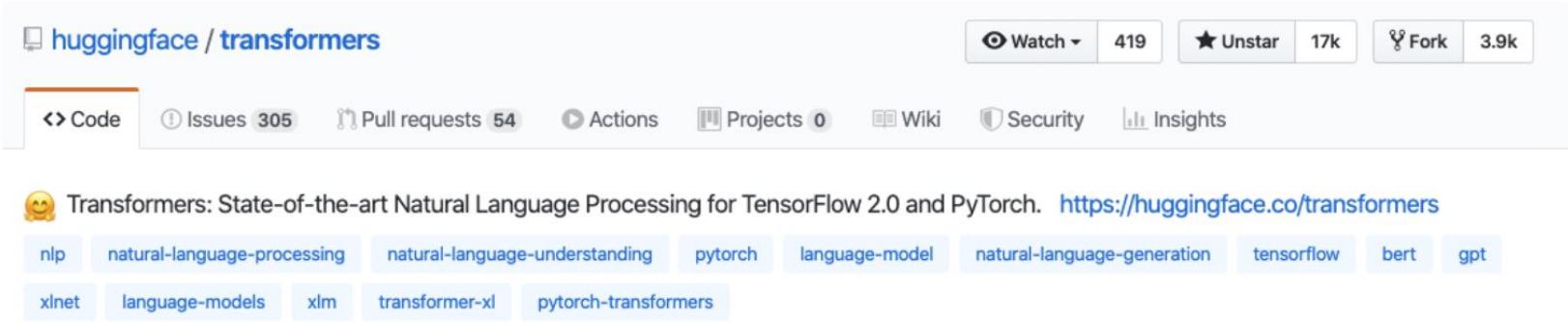
- Use sub-word embedding instead of words
  - playing → play, ##ing
- Segment tells you which sentence the word belongs
- Trained 40 epoches on Wikipedia (2.5B tokens) + BookCorpus (0.8B tokens)
- Two releases: BERT-base, BERT-large

Use BERT in Practice

- **TensorFlow**: https://github.com/google-research/bert



- **PyTorch**: https://github.com/huggingface/transformers



BERT IS VERY STRONG FOR MANY TASKS

BiLSTM: 63.9

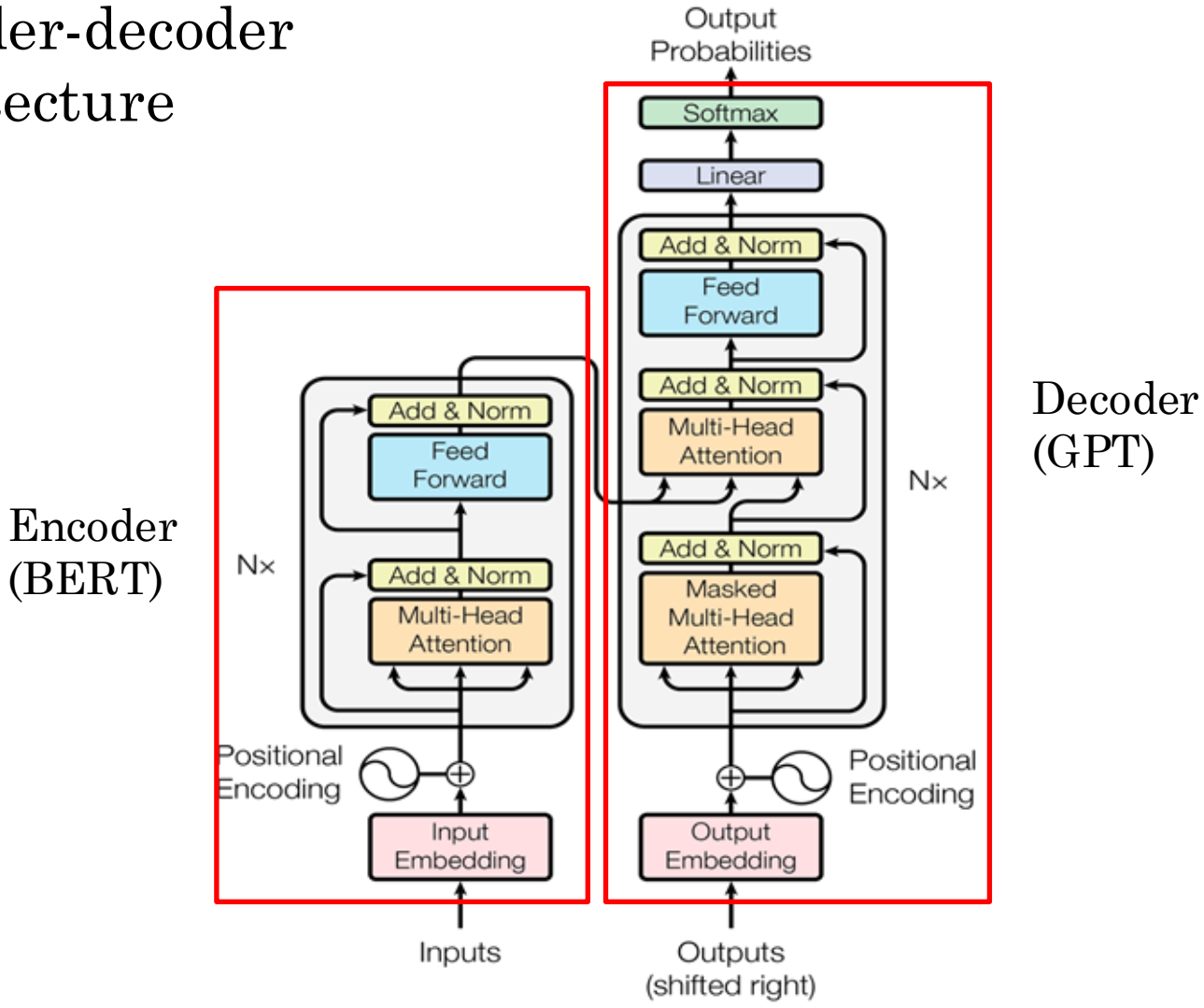
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

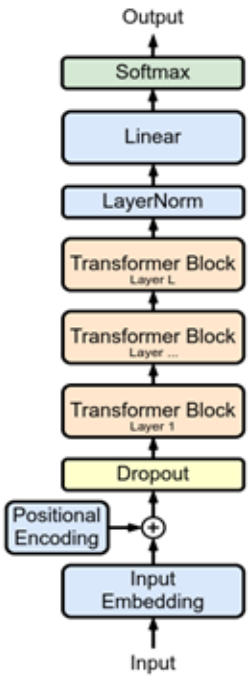
RoBERTa and XLNet are optimized versions of BERT with different pretraining approach. Archi is the same!

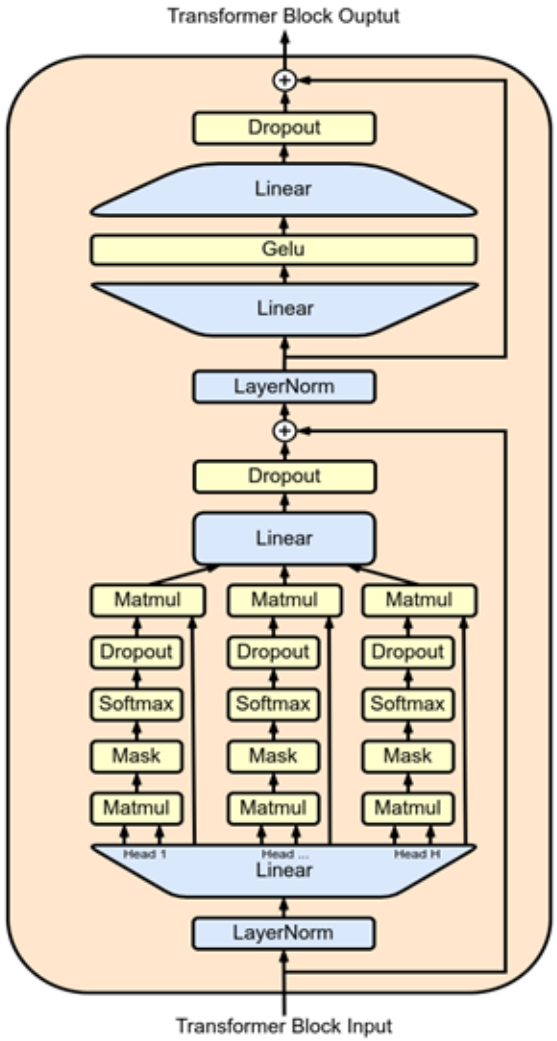
The Real Transformer (T5)

- Encoder-decoder architecture



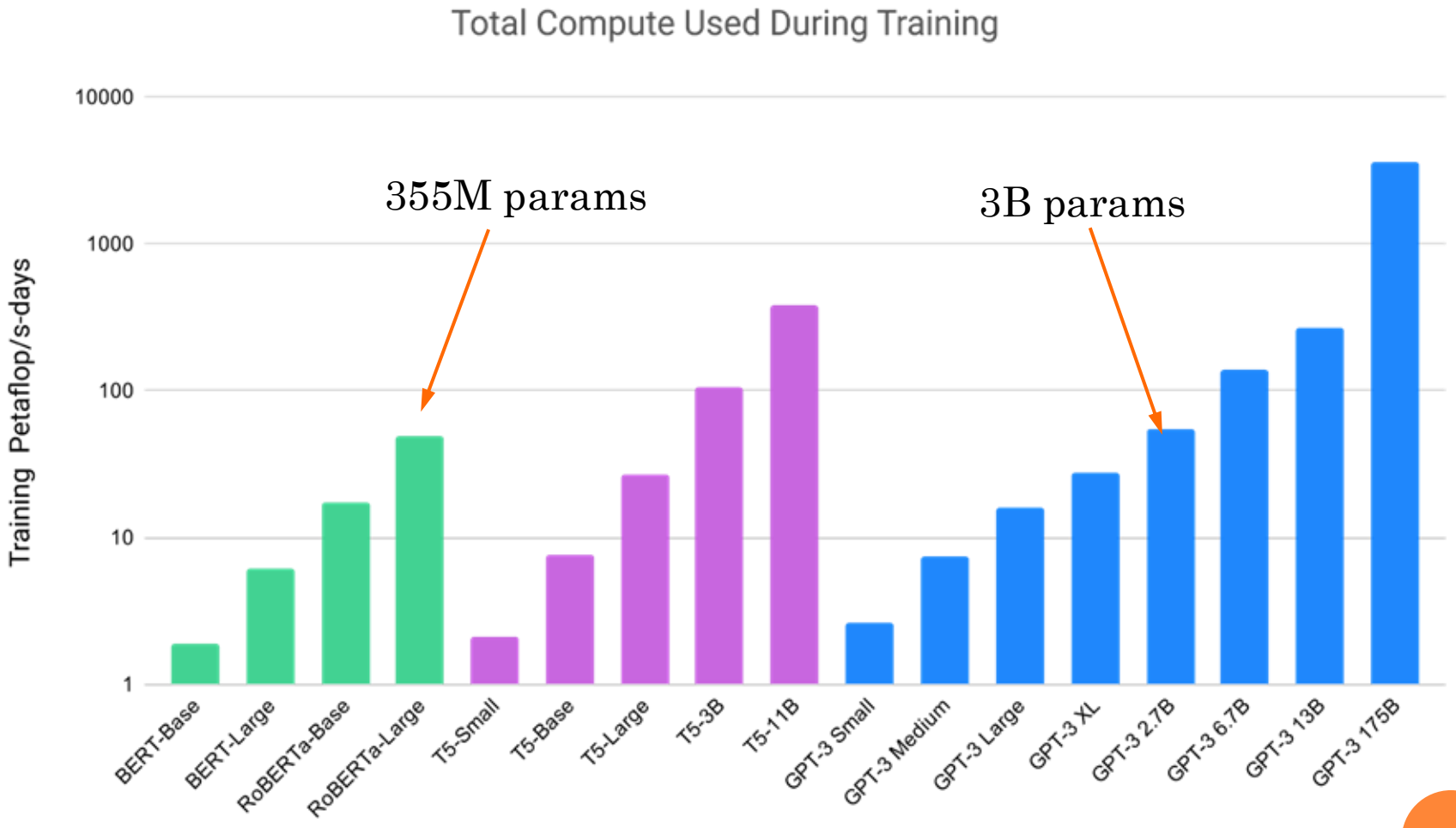
GPT-3 Architecture





- Decoder only
- Objective is to predict next token given previous tokens
- Up to 96 transformer blocks
- Each with 96 attention heads
- Up to 175B params
- Pretrained on 570GB of internet data
- Good for few-shot learning at inference

Cost of Training GPT-3



Training more params on fewer tokens (Pre-Chinchilla paradigm)

Quiz: Training a smaller model longer (1 pt)

- Subsequent research showed that it's more beneficial to train a model with fewer parameters, but on much more data. Why? (Select all that apply)

a) Pre-Chinchilla law dictates that

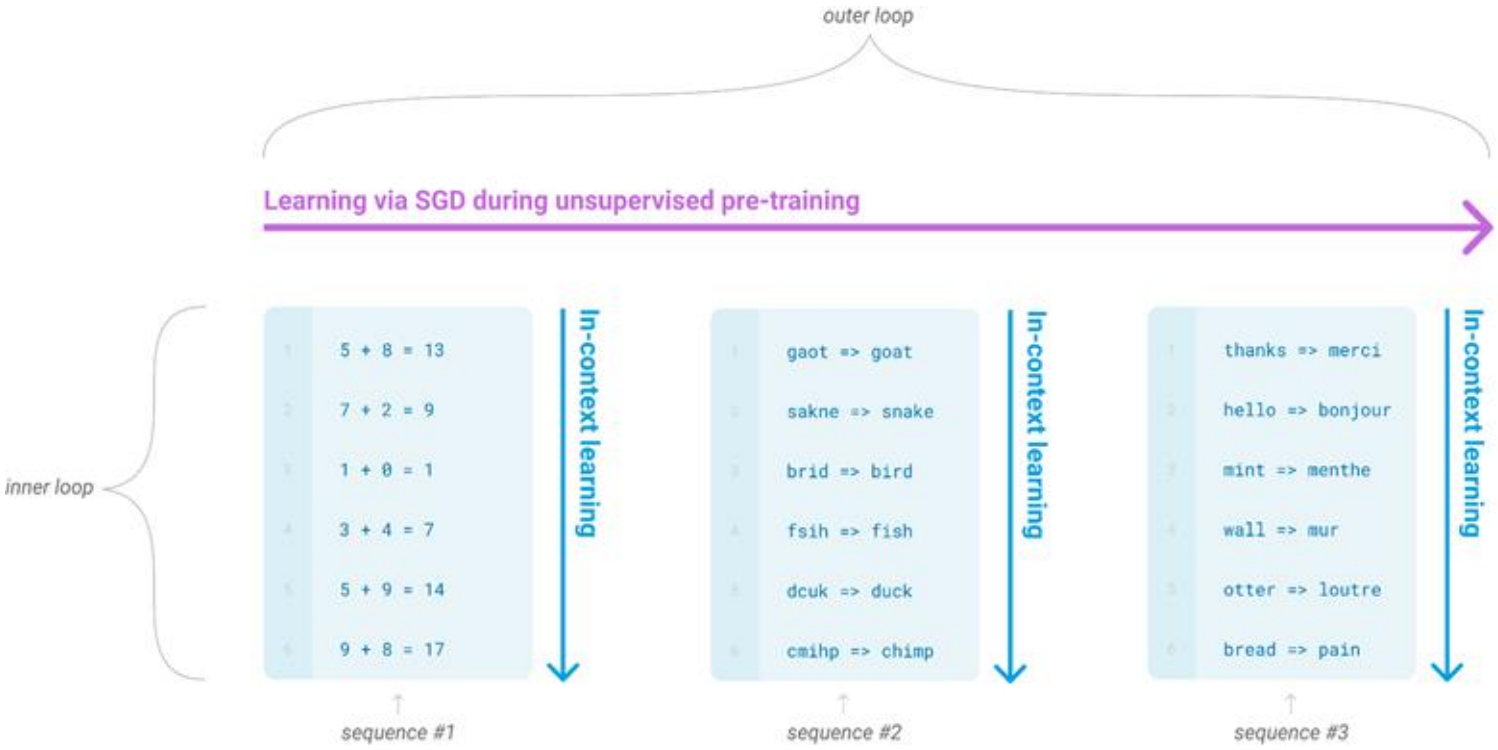
b) GPT-3 was under-trained on tokens relative to it's model size

c) Smaller models have much smaller inference cost

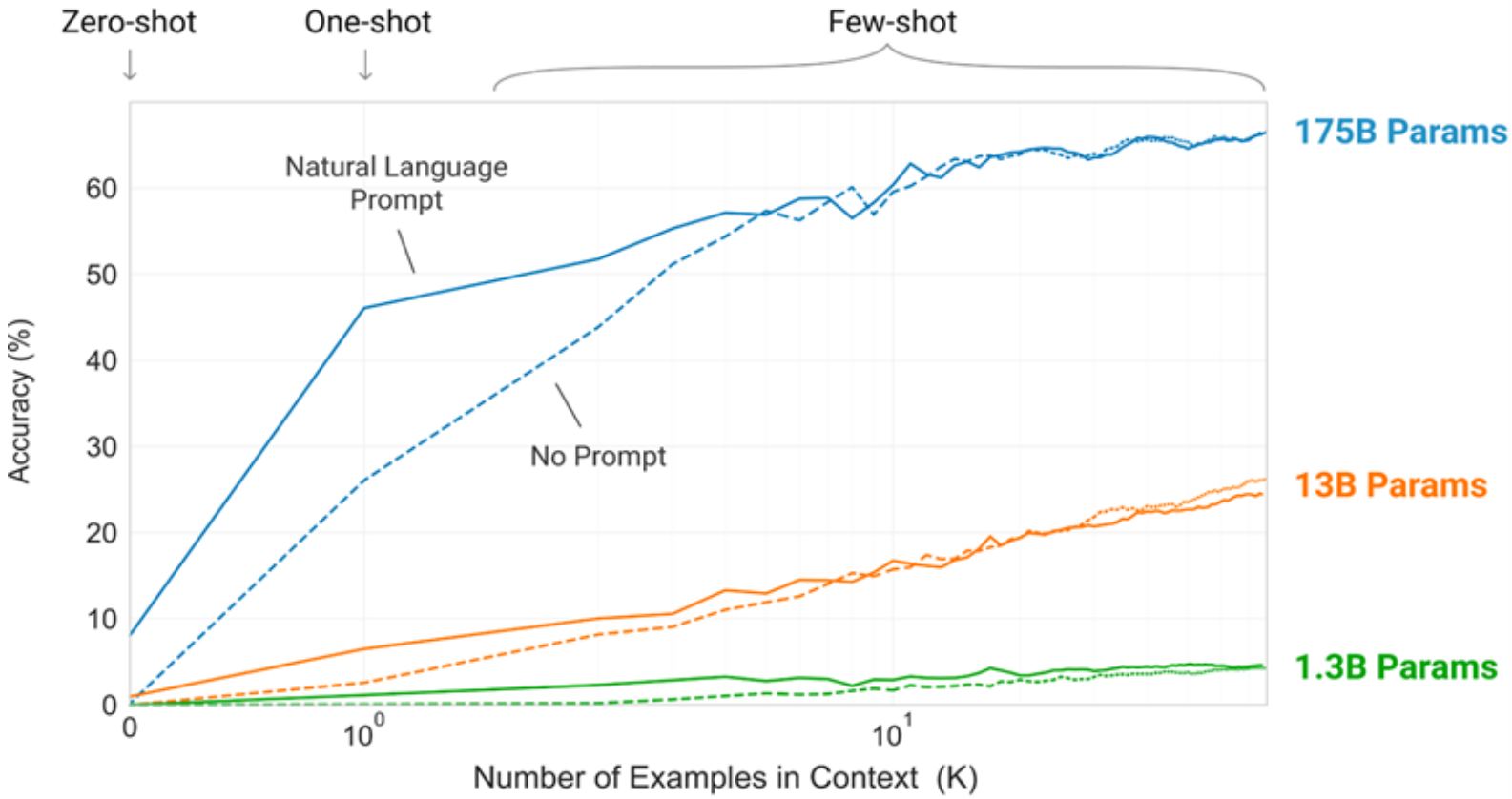
d) Training large models takes longer and is more expensive

In-context Learning

- The LLM learns a new task from a small set of examples presented within the context (the prompt) at inference time (**params frozen**).
- The key idea is to learn from analogy

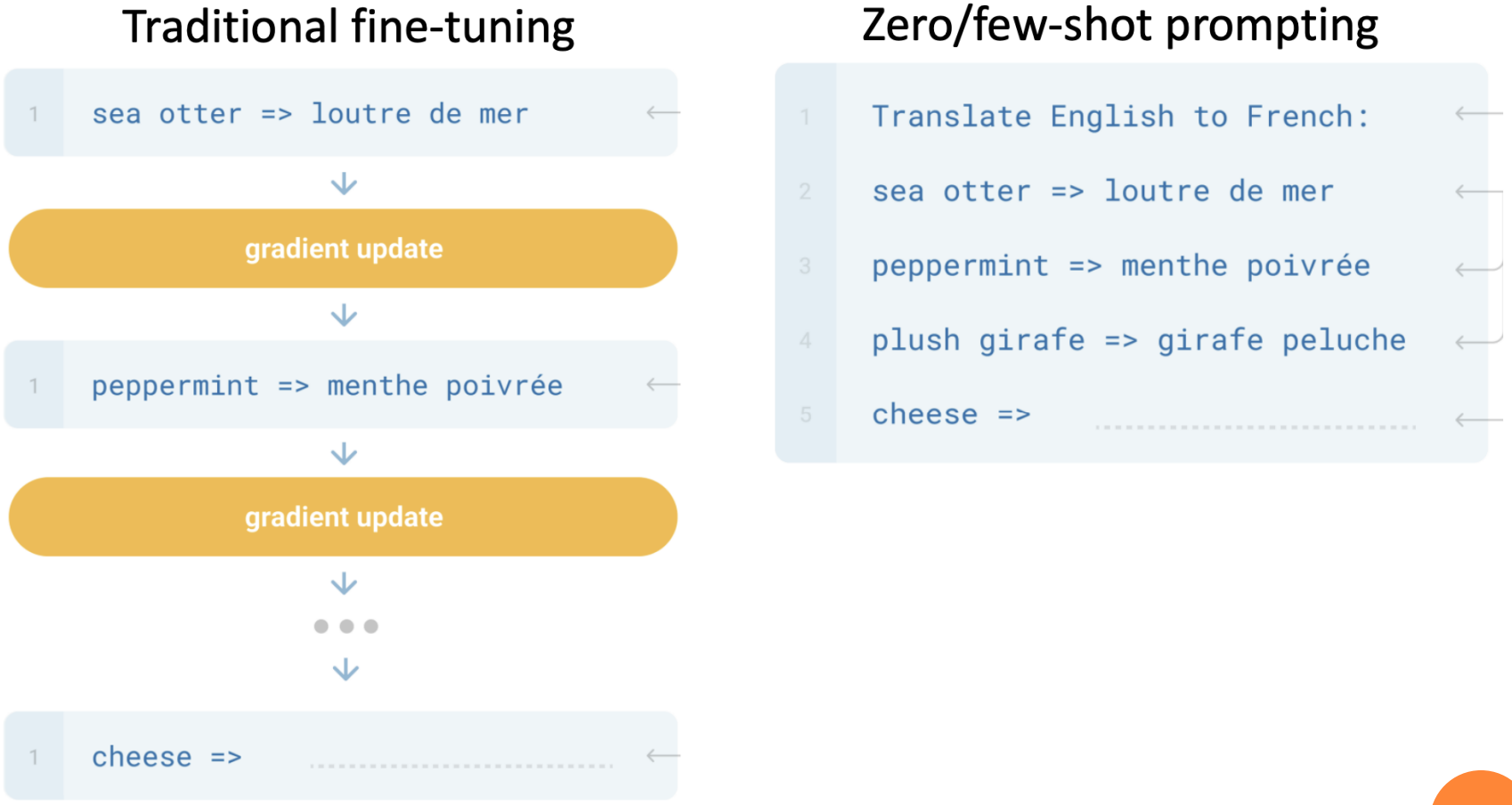


In-Context Learning Curve of GPT-3



Larger models make increasingly efficient use of in-context information

Prompting the LLMs



Limitation of Prompts

- Some tasks seem too hard for even large LMs to learn through prompting alone.
- Especially tasks involving richer, multi-step reasoning. (Humans struggle at these tasks too!)

```
19583 + 29534 = 49117
98394 + 49384 = 147778
29382 + 12347 = 41729
93847 + 39299 = ?
```

Change the prompt!

Chain-of-Thought Prompting

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. ✓

Zero-shot Chain-of-thought Prompting

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. ✓

We don't even need to present the reasoning to LLM!

Zero-shot CoT Trigger Prompt	Accuracy
Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
Let's think step by step. (*1)	78.7
First, (*2)	77.3
Let's think about this logically.	74.5
Let's solve this problem by splitting it into steps. (*3)	72.2
Let's be realistic and think step by step.	70.8
Let's think like a detective step by step.	70.3
Let's think	57.5
Before we dive into the answer,	55.7
The answer is after the proof.	45.7
(Zero-shot)	17.7

This is possible due to "emergent" capability of LLM!

Instruction Tuning

- Language models are not aligned with user intent [Ouyang et al., 2022]:

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

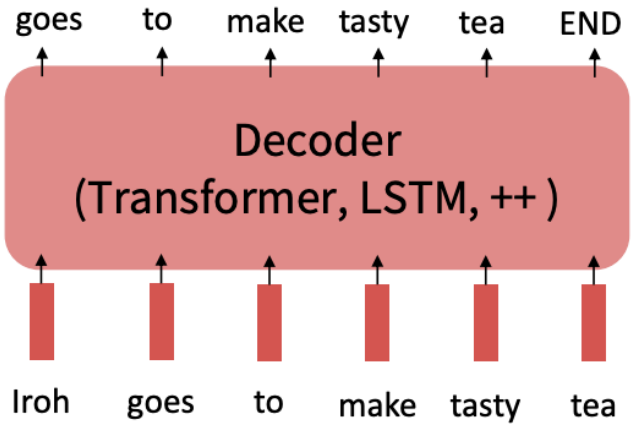
COMPLETION **Human**

A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

Traditional Pretrain-finetune paradigm

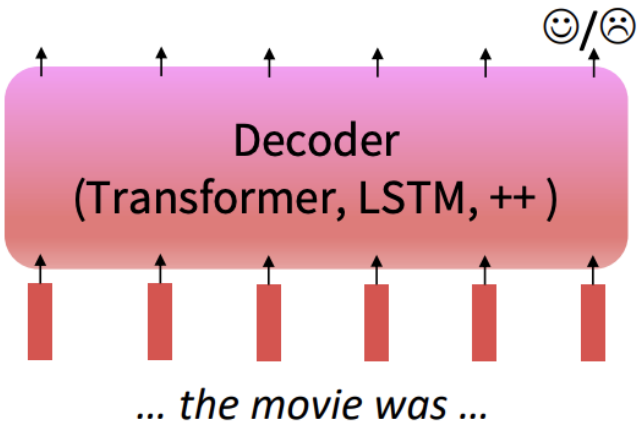
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

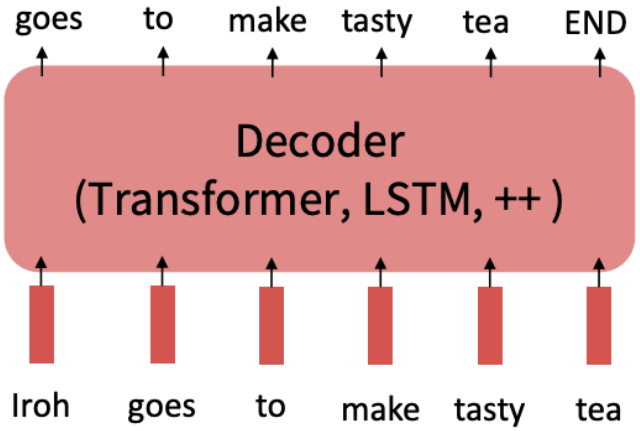
Not many labels; adapt to the task!



Scaling up the finetuning

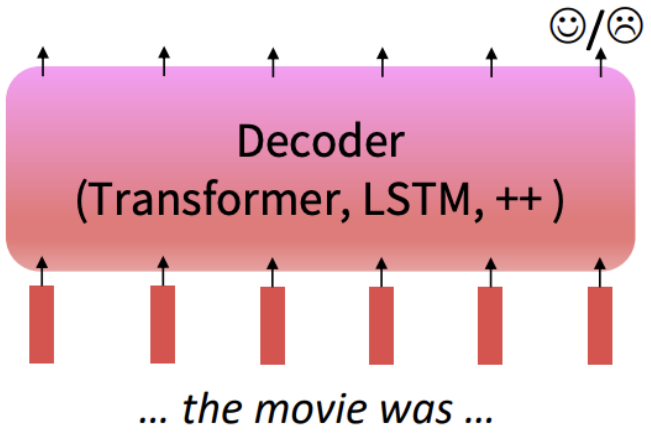
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



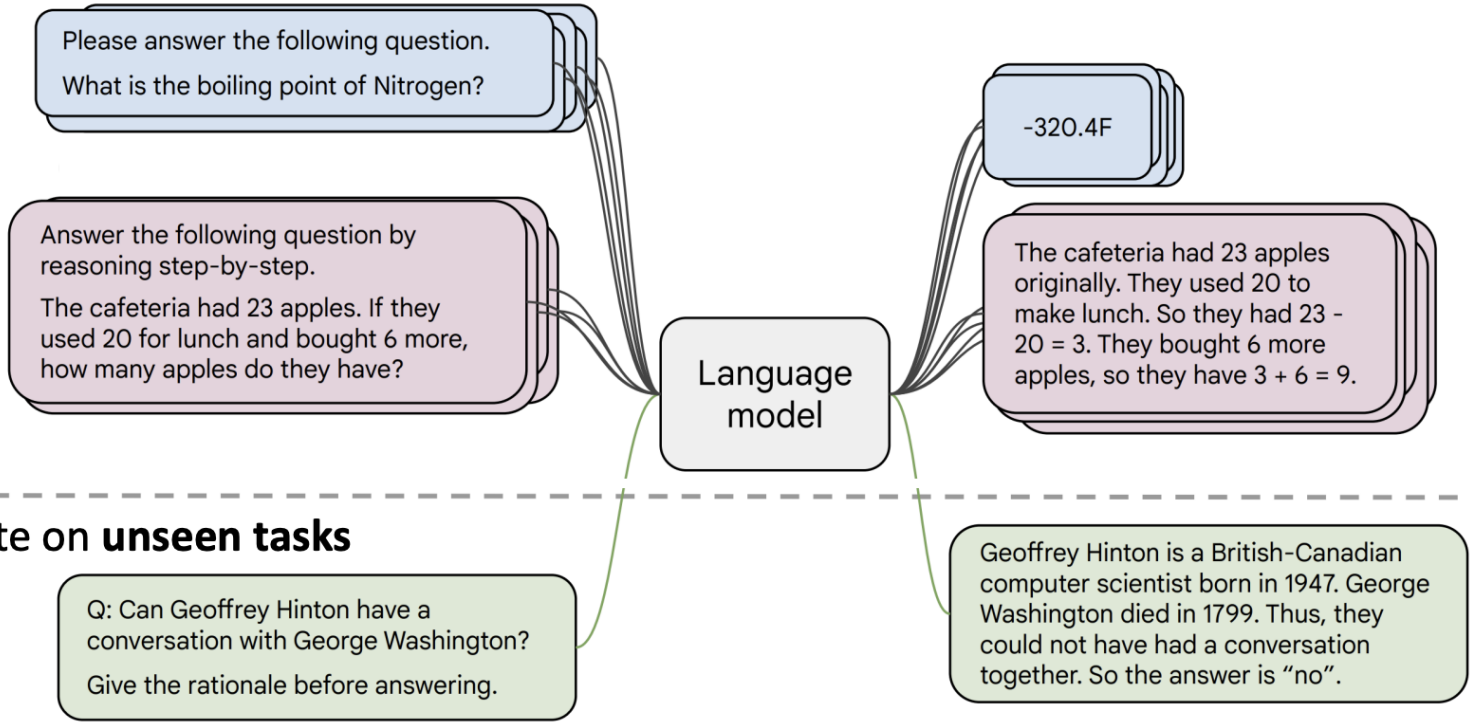
Step 2: Finetune (on many tasks)

~~Not~~ many labels; adapt to the tasks!



Instruction Tuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



- Evaluate on **unseen tasks**

Quiz: LLM getting smarter? (3 pts)

- LLMs are pretrained on massive amount of human text (or knowledge). As they interact with humans, they continue to learn new knowledge. Given to prevalent use of the LLMs online, do you think LLMs like ChatGPT or Gemini will give better and more accurate answers in 5 to 10 years? Please explain your answer.