CSE 4392 Special Topics
Natural Language Processing

Transformer and Large Language Models

2024 Spring

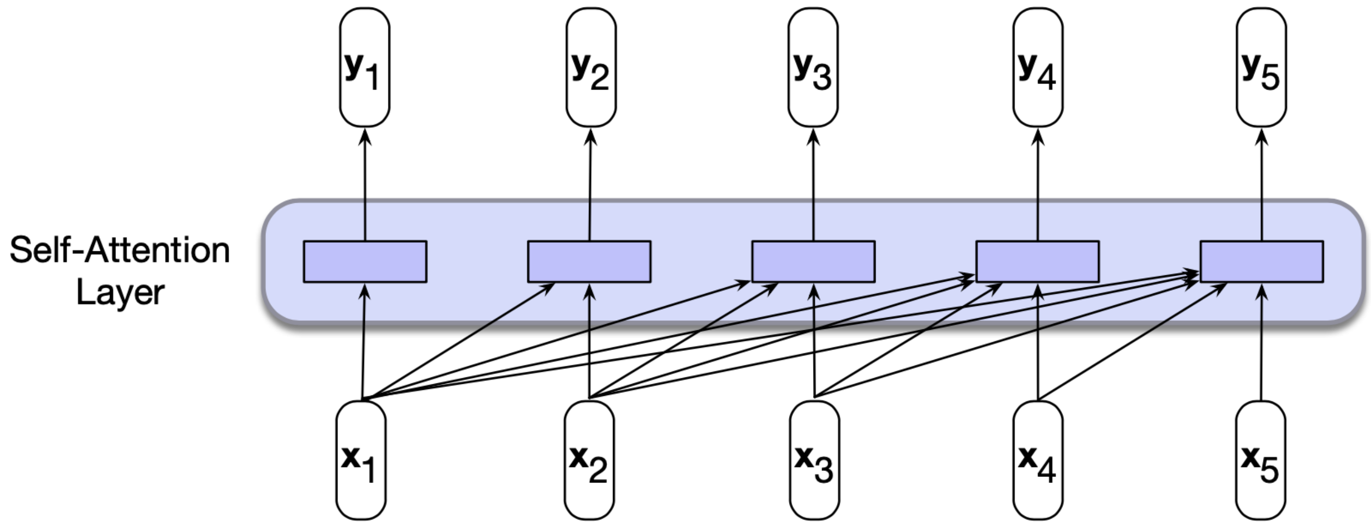
PRETRAINING

- 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

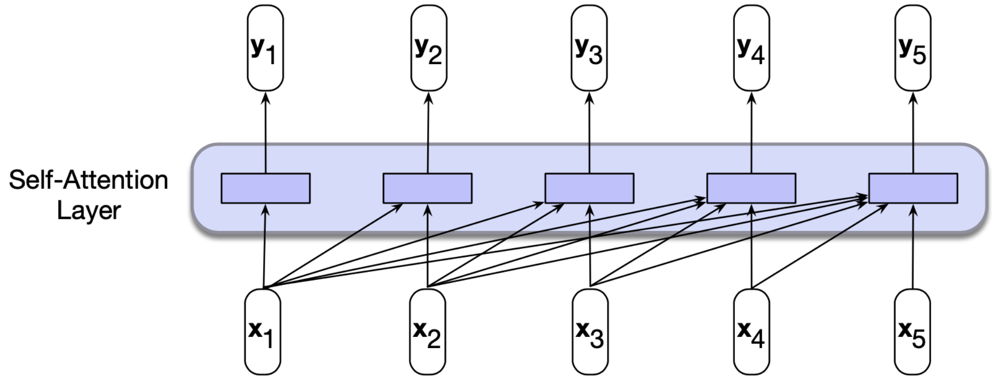
- Like LSTM, transformers can handle long-range dependencies
- But it doesn't use recurrent connections
  - recurrent architectures are hard to parallelize
  - transformers can be parallelized and are more efficient
- Two main ideas in transformers:
  - **Self-attention**
  - **Positional embeddings**

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 $x_1, \ldots, x_i$, 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;

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

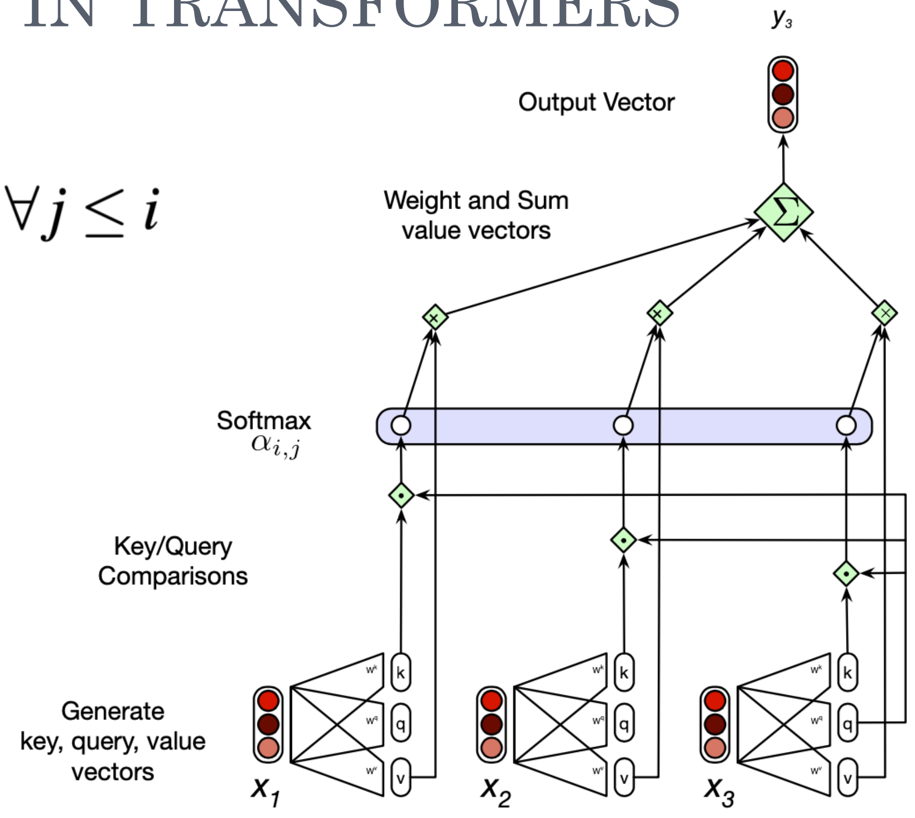
- $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are vectors of d-dimension.
- $W^Q \in \mathbb{R}^{d\times d}, W^K \in \mathbb{R}^{d\times d}, W^V \in \mathbb{R}^{d\times d}$

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:

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

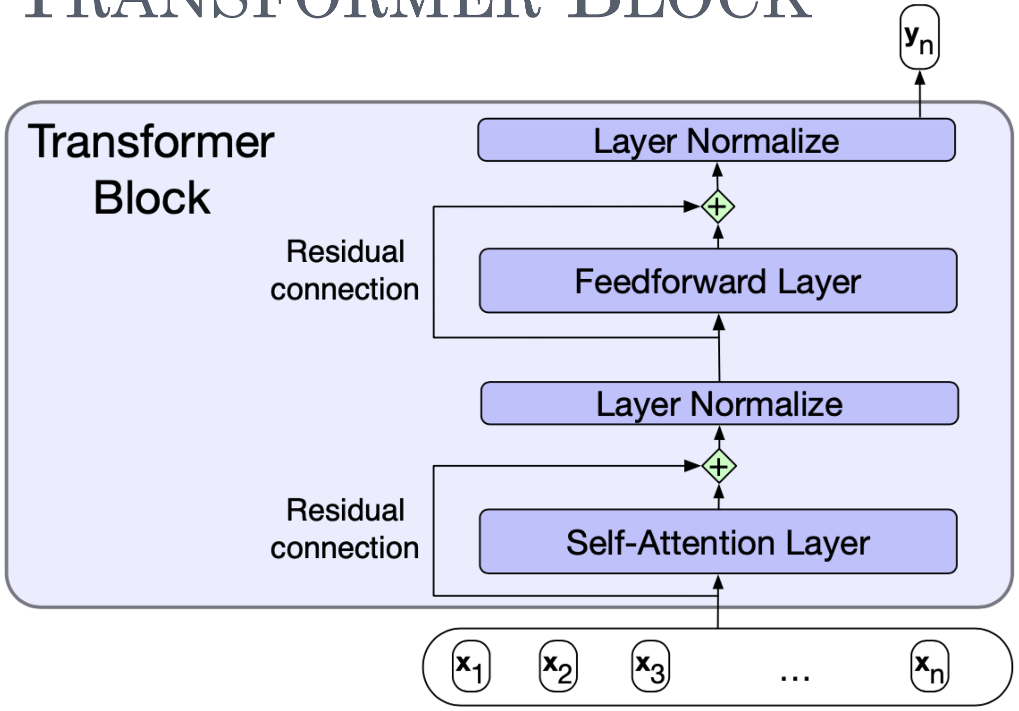
Computing all inputs together:

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

Quiz: Complexity of Attention

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

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

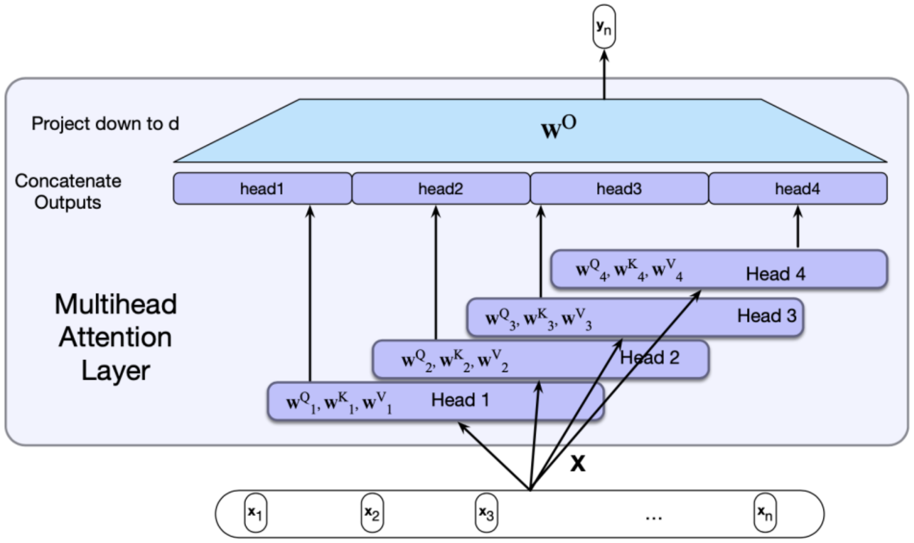
- Layer normalization can be any normalization that keeps the values of hidden layers in a range that is "gradient friendly."
- Mean: $$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i$$
- Standard dev: $$\sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i-\mu)^2}$$
- normalized: $$\hat{\mathbf{x}} = \frac{(\mathbf{x}-\mu)}{\sigma}$$

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

trainable params

Multi-head Attention

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



y_n
Project down to d
w^O
Concatenate Outputs
head1
head2
head3
head4
Multihead Attention Layer
w^Q_4, w^K_4, w^V_4 Head 4
w^Q_3, w^K_3, w^V_3 Head 3
w^Q_2, w^K_2, w^V_2 Head 2
w^Q_1, w^K_1, w^V_1 Head 1
X
x_1 x_2 x_3 … x_n

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

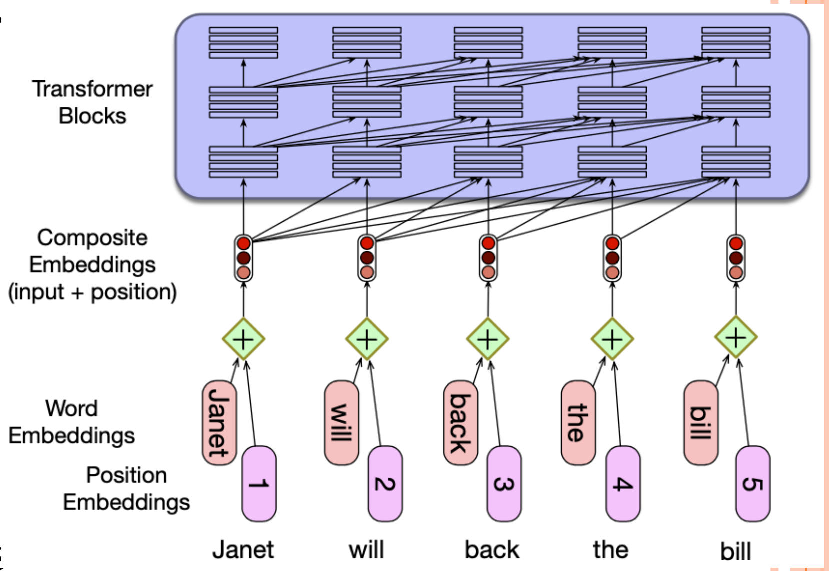
$$\text{MultiHeadAttention}(\mathbf{X}) = (\mathbf{head}_1 \oplus \mathbf{head}_2 \ldots \oplus \mathbf{head}_h)\mathbf{W}^O$$

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

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

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

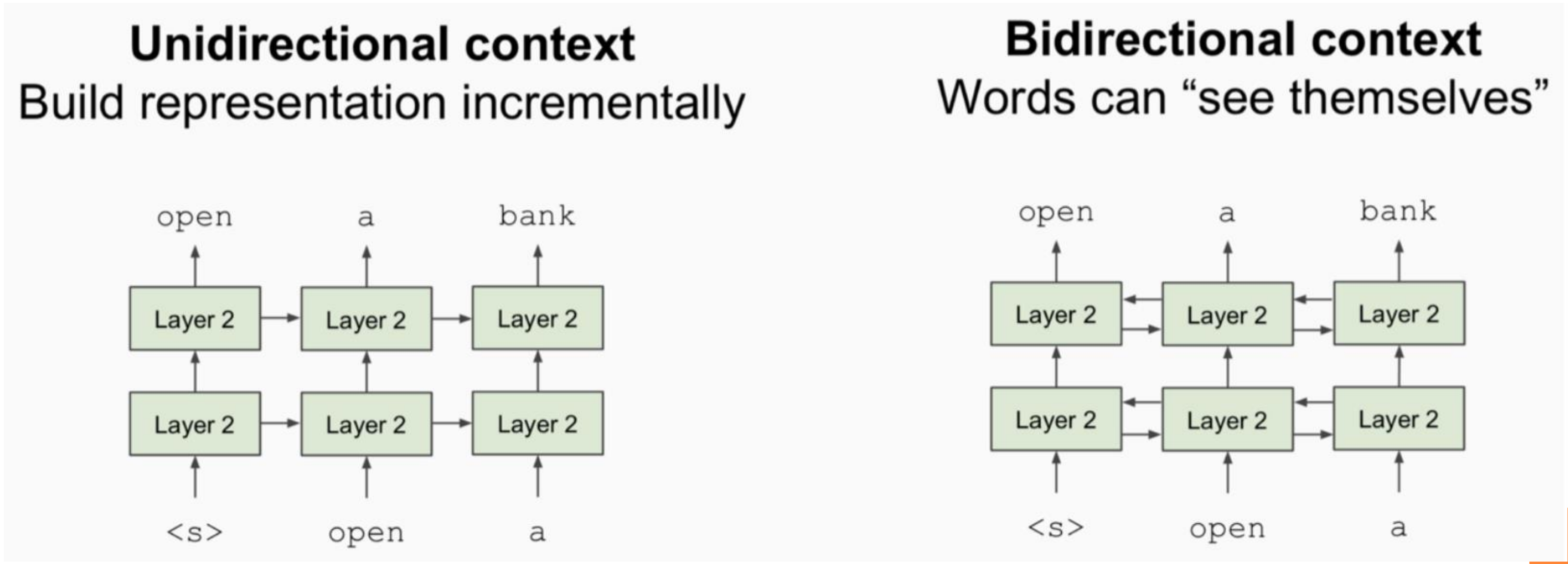
- Why do we concatenate the multipile head vectors together to get the overall attention, but simply add (element-wise) the word vector and positional vector to create the new input vector?

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*

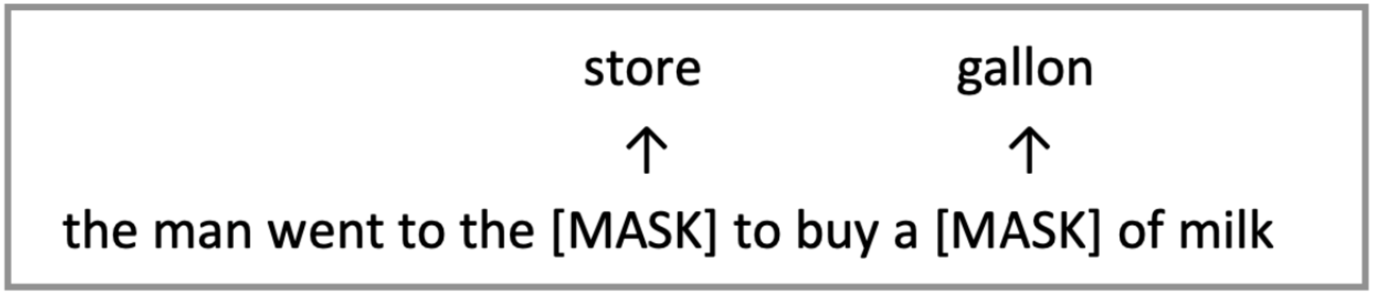
BIDIRECTIONAL ENCODERS

- 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

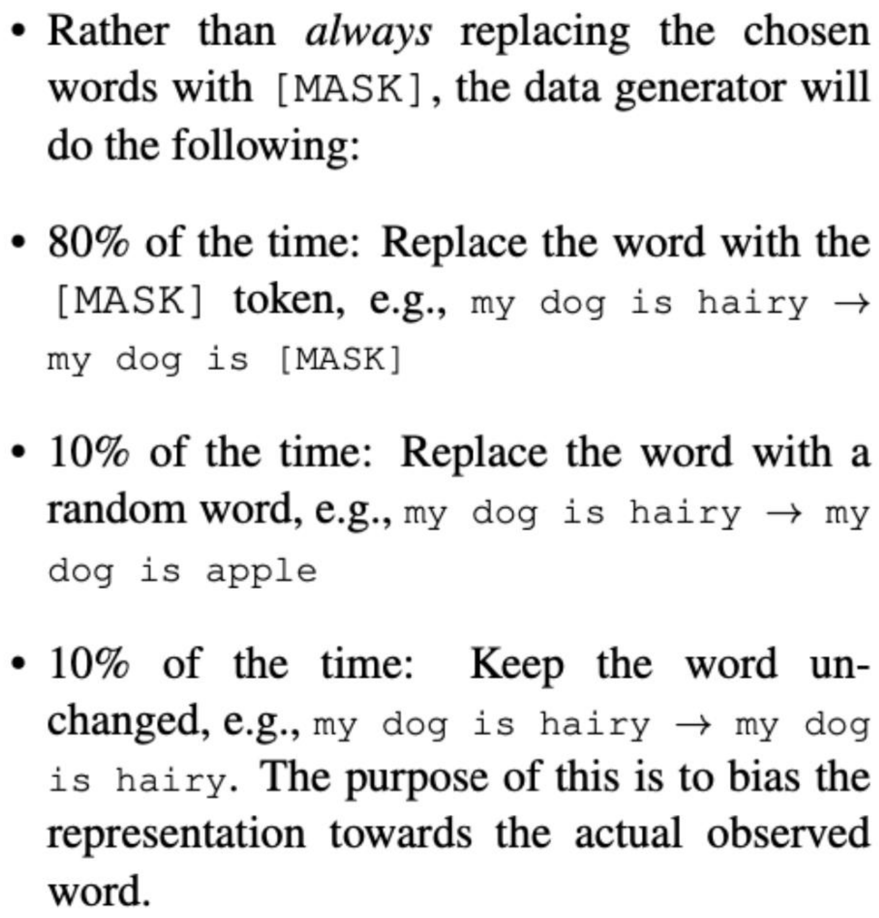
- How to pretrain the language model?
- Solution: Mask out 15% of the input words, and then predict the masked words



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

Masked Language Models

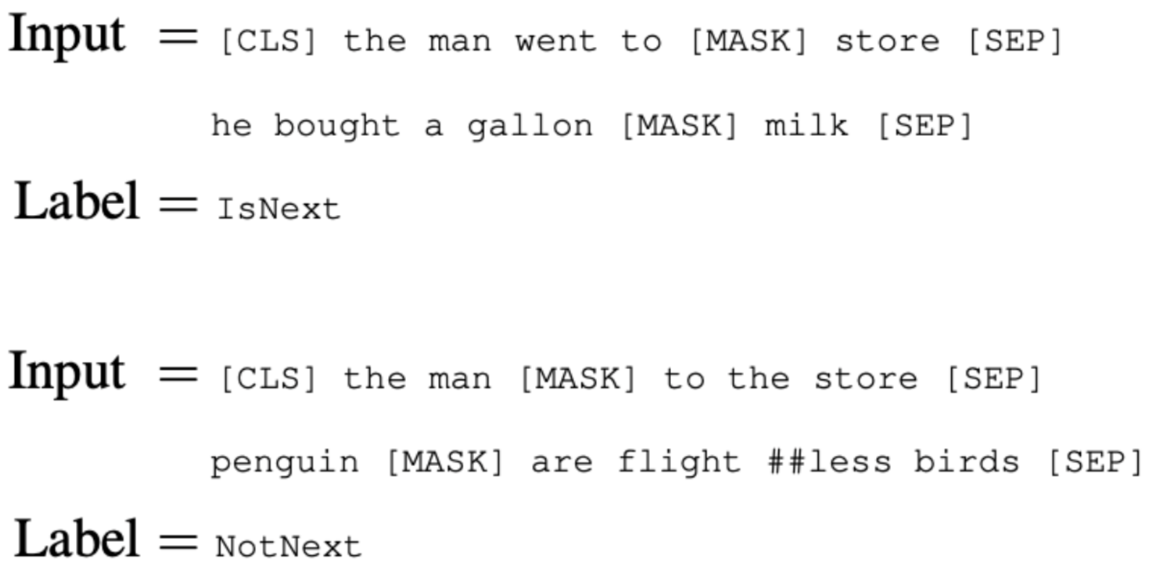
- Because BERT will never see [MASK] in real-world data, training data is a little more complicated:



- 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 is followed after 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]`

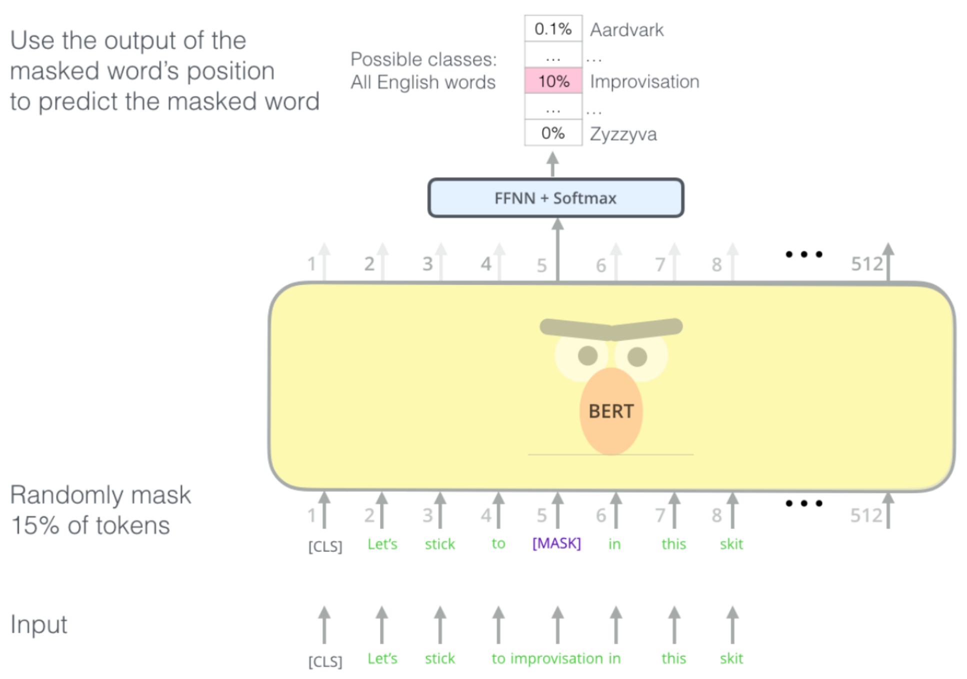
Label = `NotNext`

- Recent paper shows that NSP is not necessary…

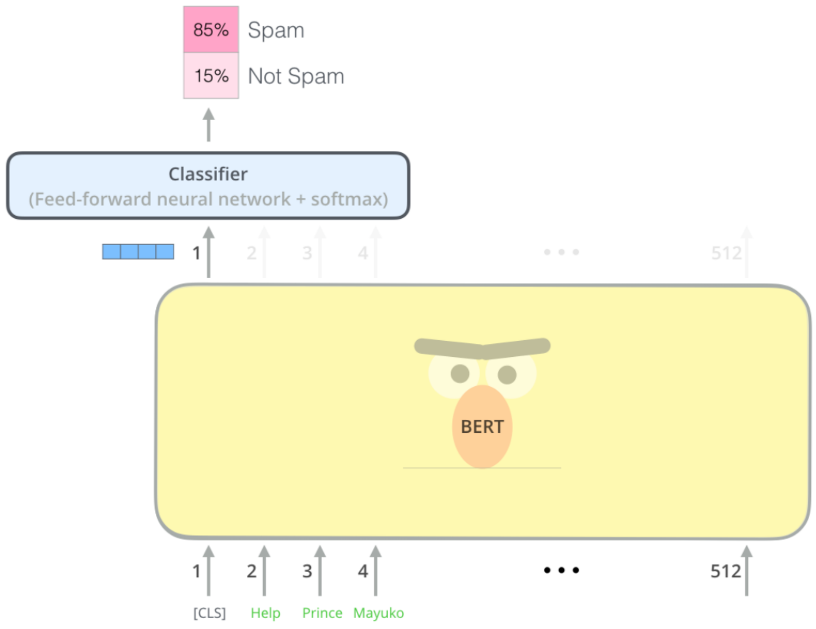
(Joshi*, Chen* et al, 2019) :SpanBERT: Improving Pre-training by Representing and Predicting Spans
(Liu et al, 2019): RoBERTa: A Robustly Optimized BERT Pretraining Approach

Pretraining and Fine-tuning

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

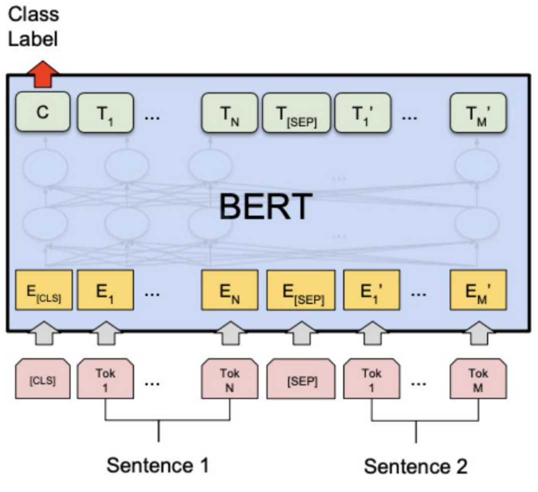

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
10% Improvisation
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

Pretrain

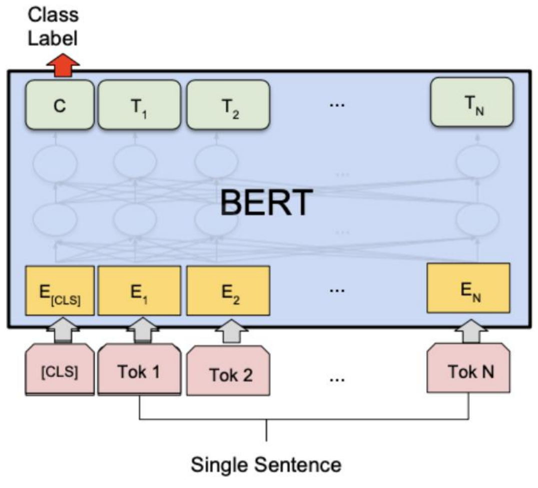


Fine-tune

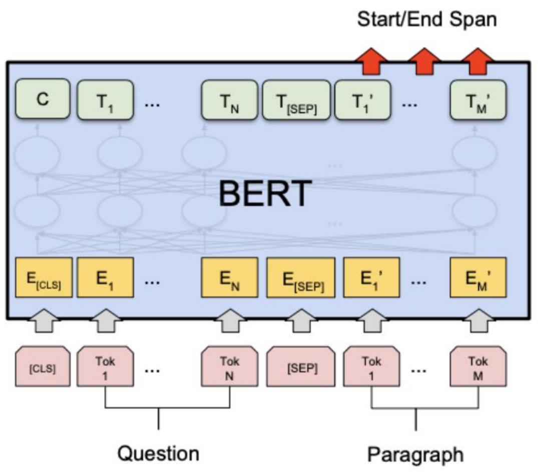
BERT Applications



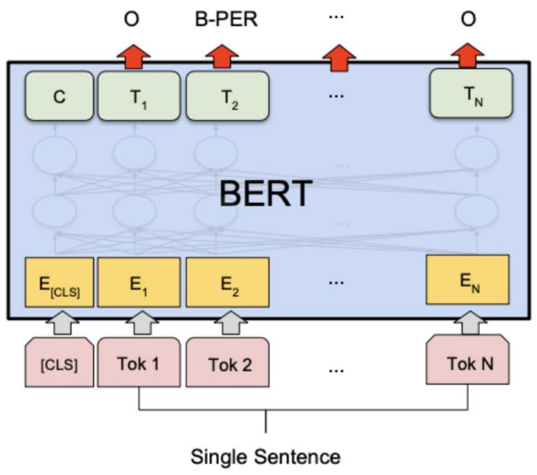
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



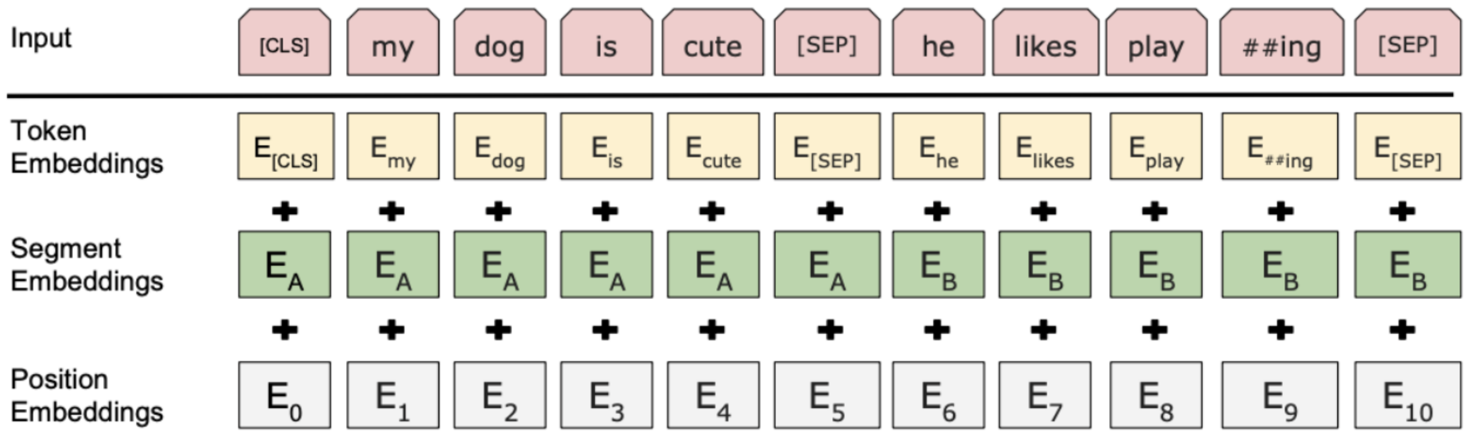
(c) Question Answering Tasks: SQuAD v1.1



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

BERT More Details

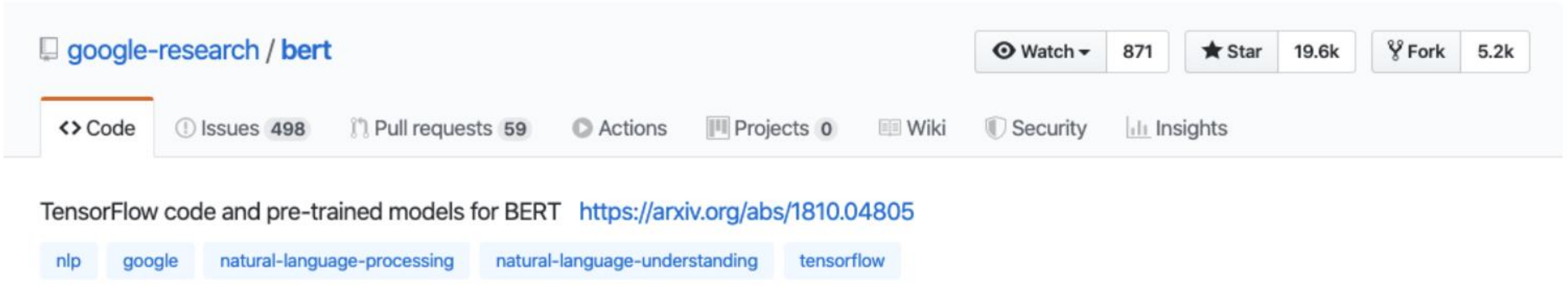
- Input representations:


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

- Use sub-word embedding instead of words
  - playing → play, ##ing
- 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



google-research / bert

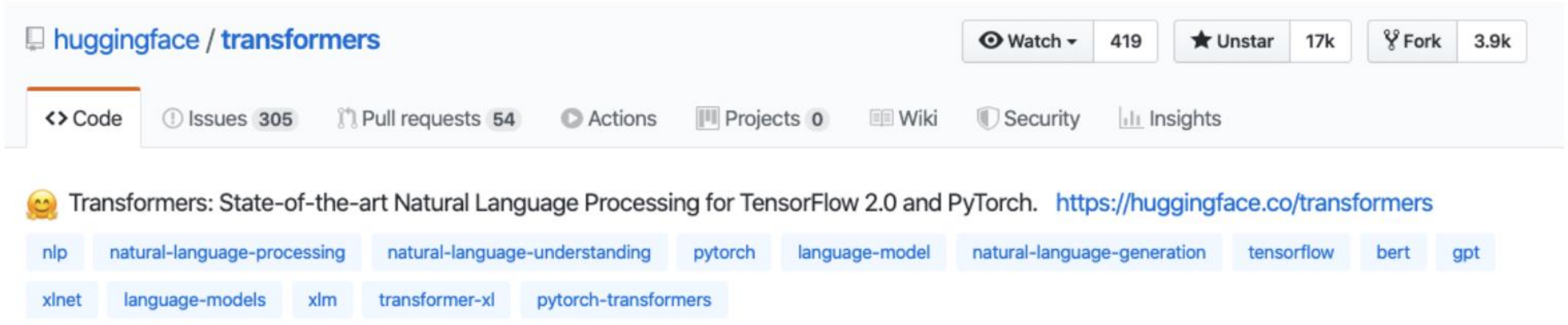
Watch 871 | Star 19.6k | Fork 5.2k

Code | Issues 498 | Pull requests 59 | Actions | Projects 0 | Wiki | Security | Insights

TensorFlow code and pre-trained models for BERT https://arxiv.org/abs/1810.04805

nlp google natural-language-processing natural-language-understanding tensorflow

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



huggingface / transformers

Watch 419 | Unstar 17k | Fork 3.9k

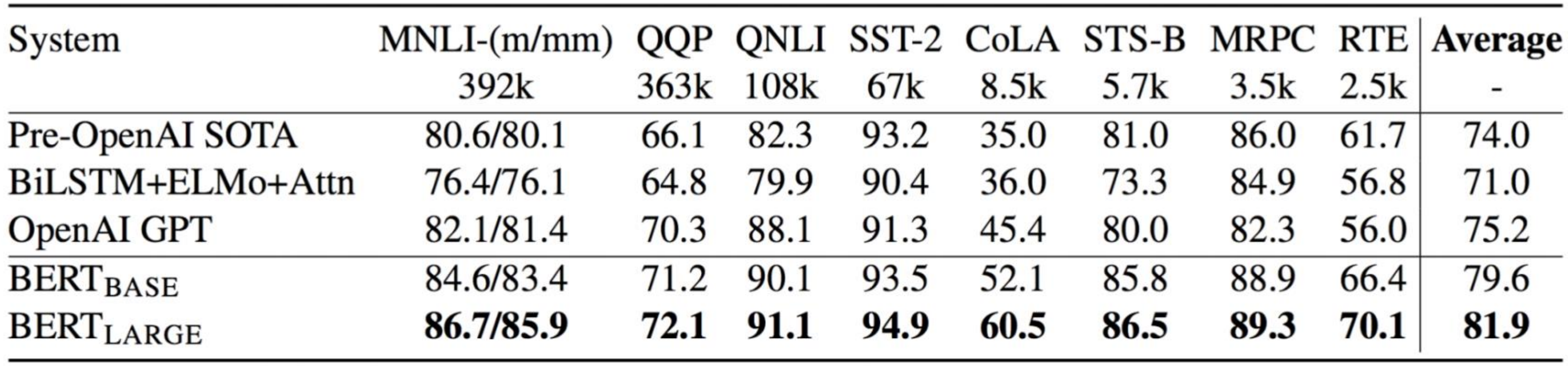
Code | Issues 305 | Pull requests 54 | Actions | Projects 0 | Wiki | Security | Insights

🤗 Transformers: State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch. https://huggingface.co/transformers

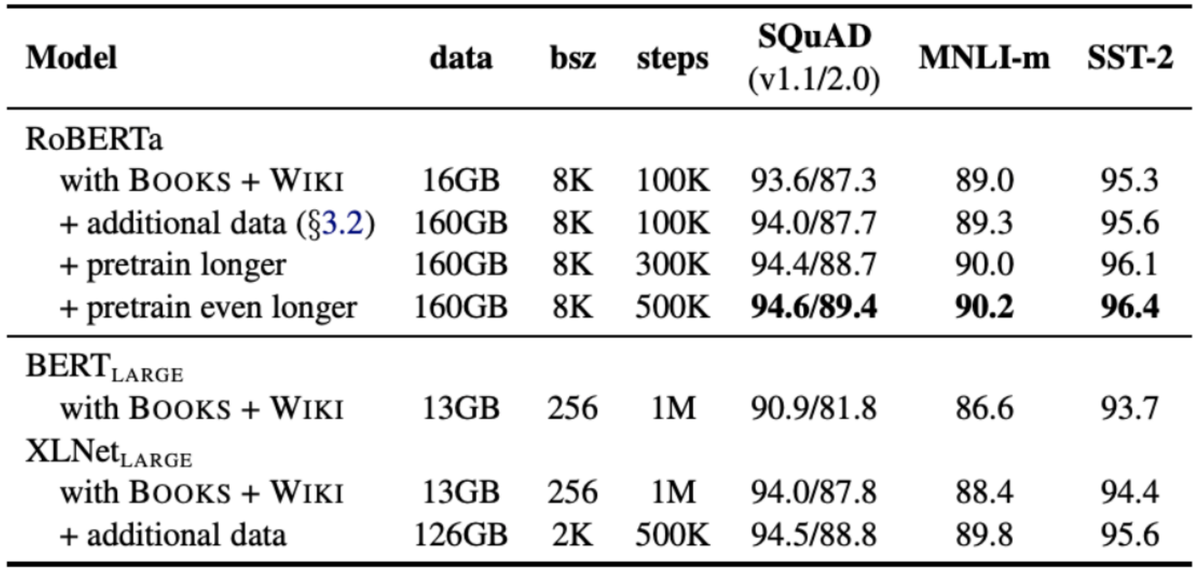
nlp natural-language-processing natural-language-understanding pytorch language-model natural-language-generation tensorflow bert gpt xlnet language-models xlm transformer-xl pytorch-transformers

BERT IS VERY STRONG FOR MANY TASKS

BiLSTM: 63.9



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

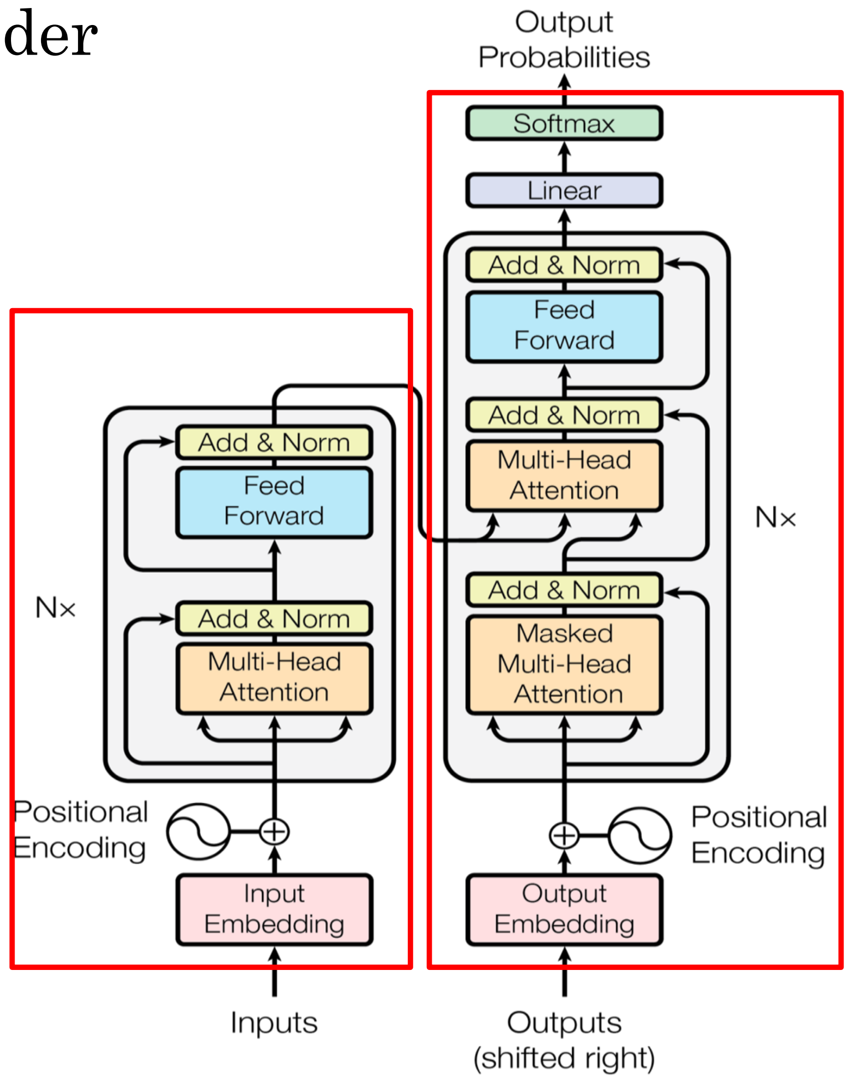


Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**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

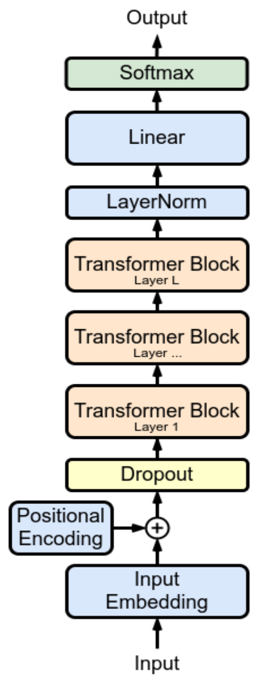
- Encoder-decoder architecture

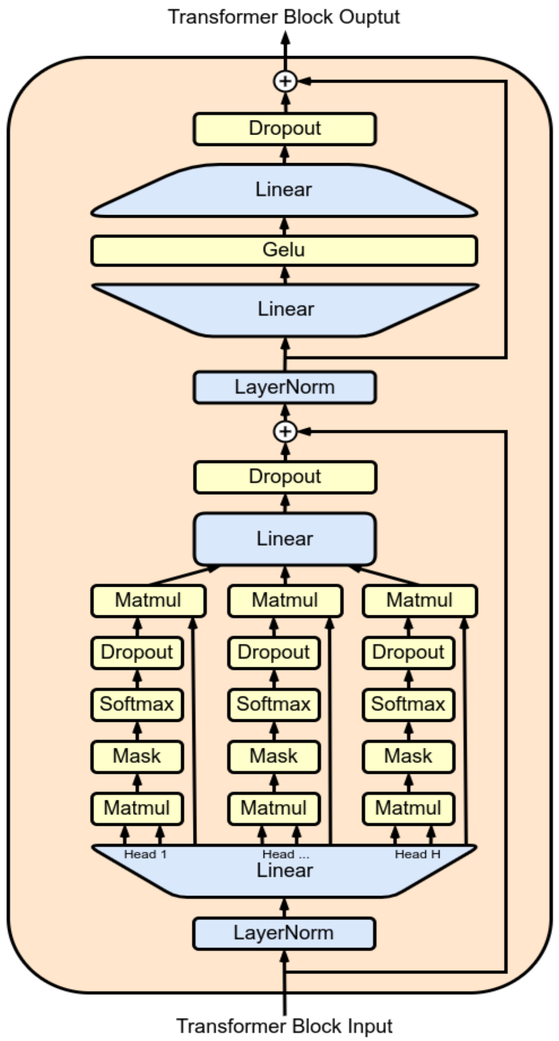


Encoder (BERT)

Decoder (GPT)

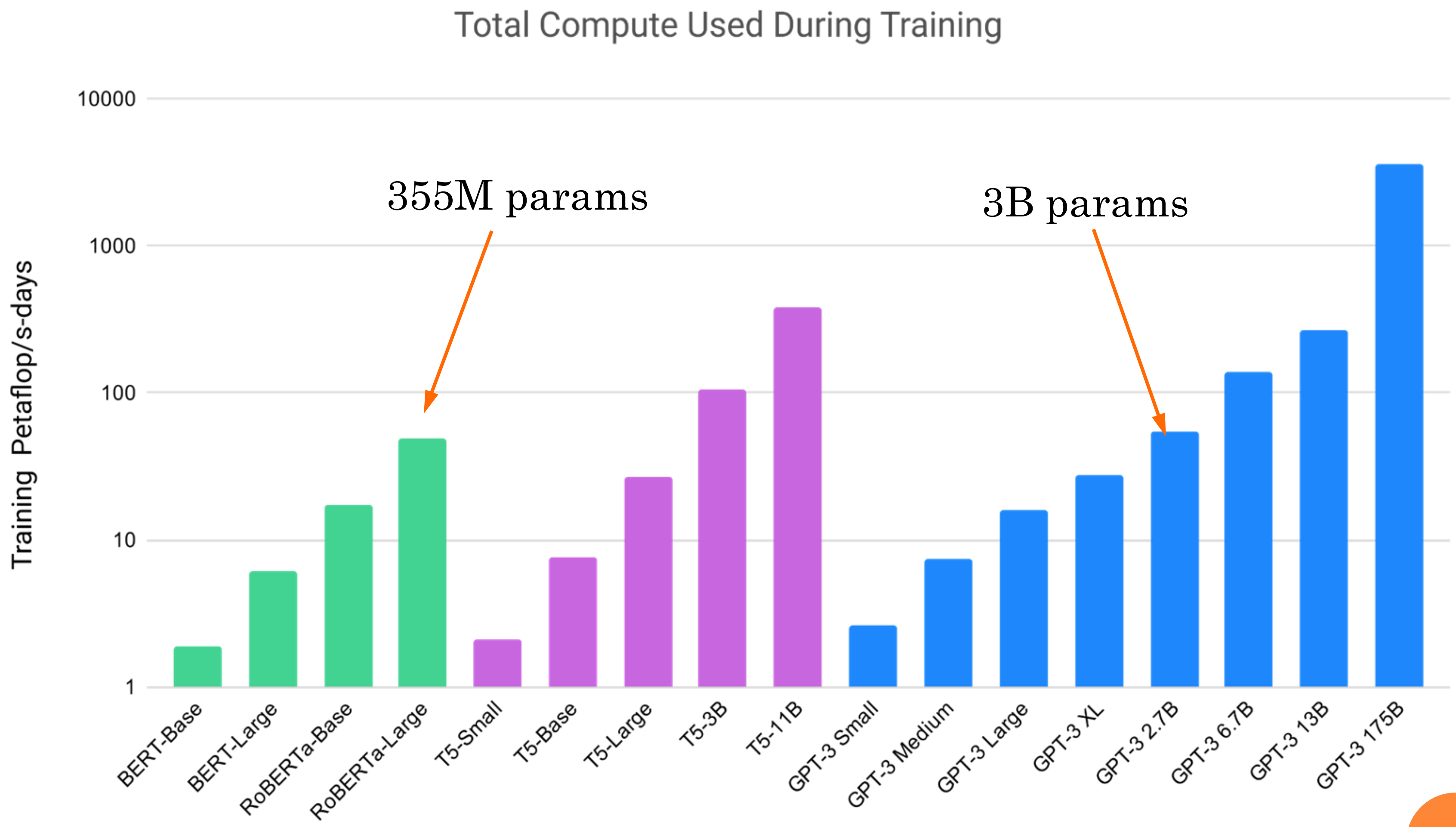
GPT-3 Architecture





- 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

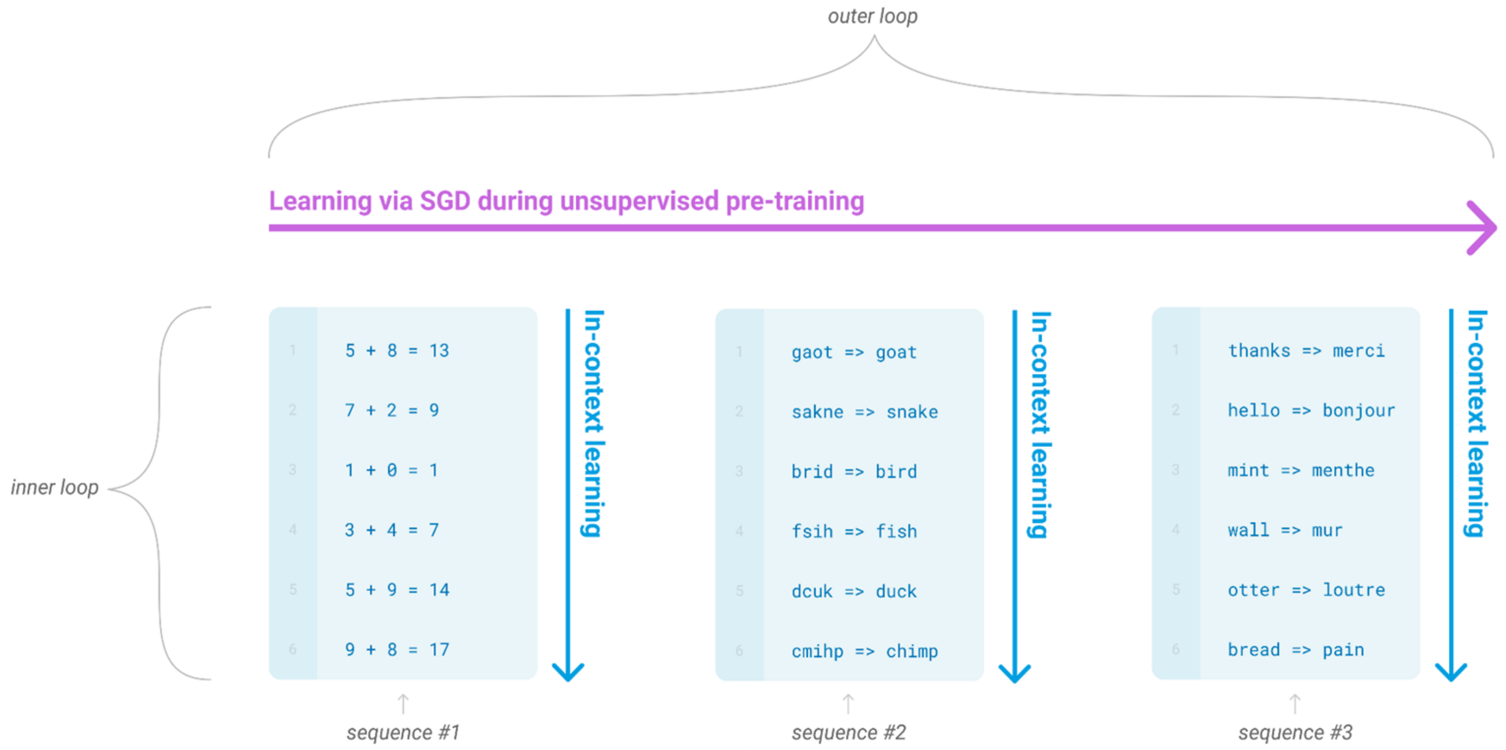
Cost of Training GPT-3



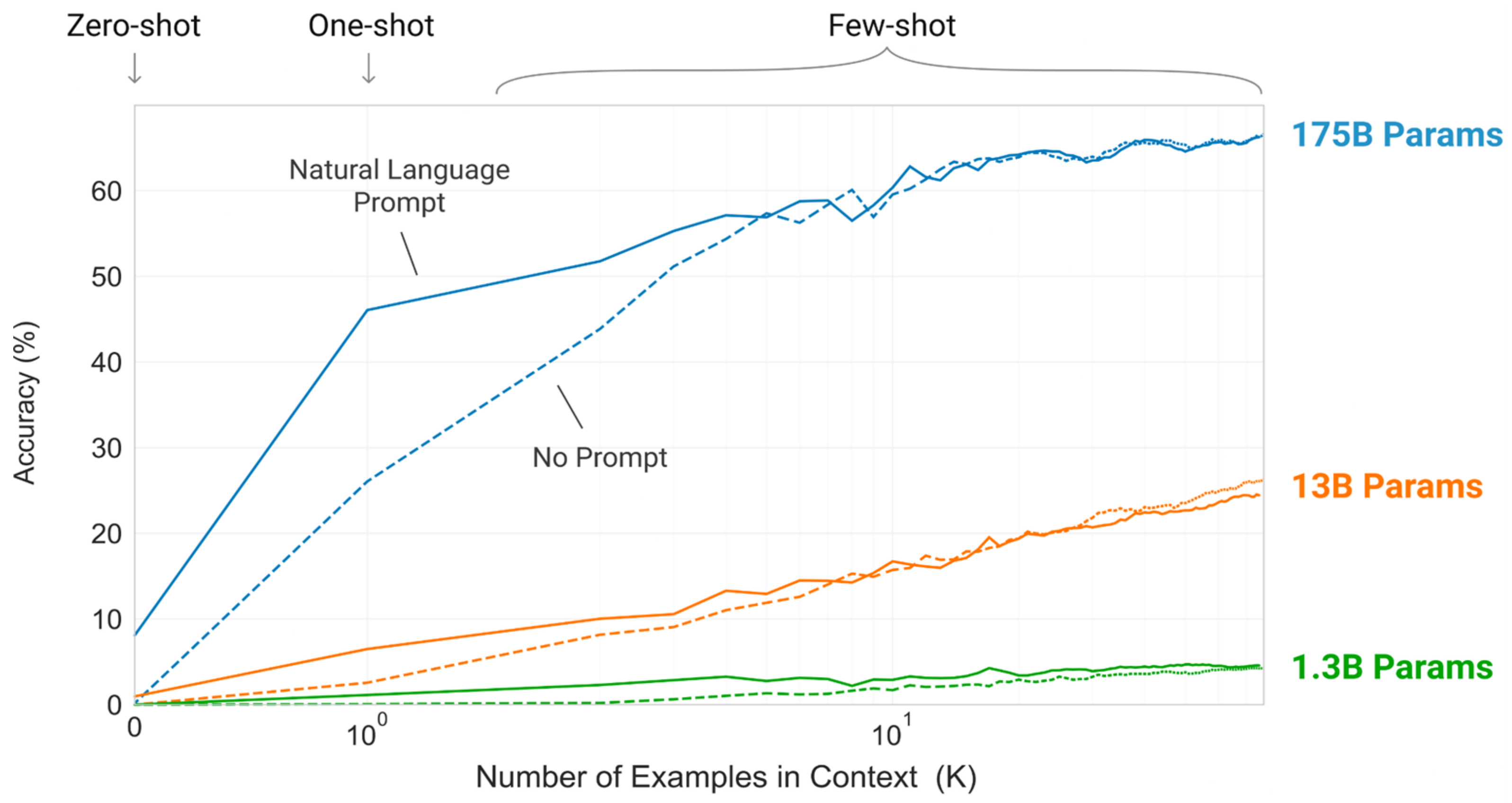
Training more params on fewer tokens

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

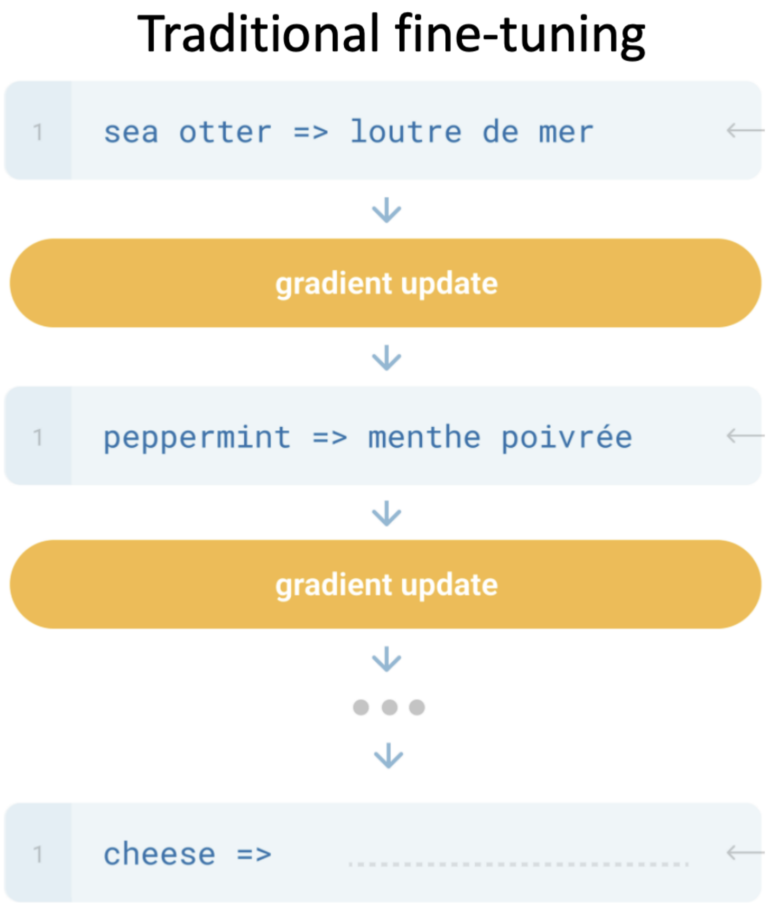


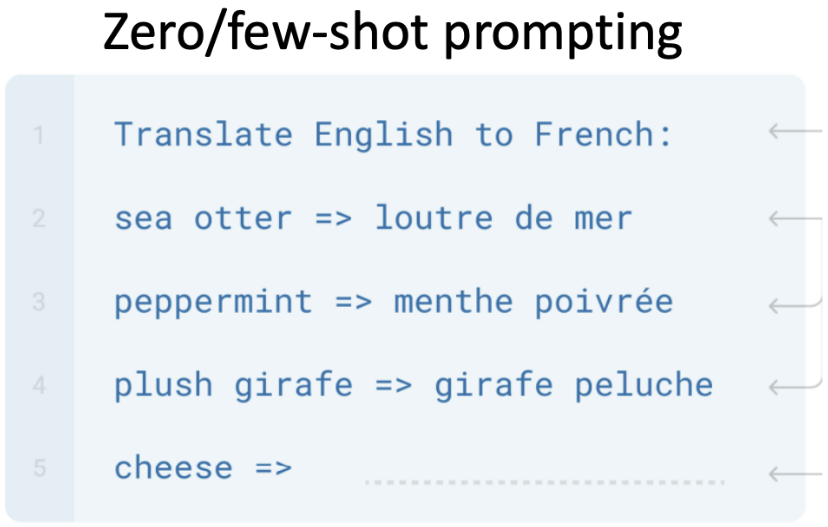
In-Context Learning Curve of GPT-3



Larger models make increasingly efficient use of in-context information

PROMPTING THE LLMS




Zero/few-shot prompting
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>

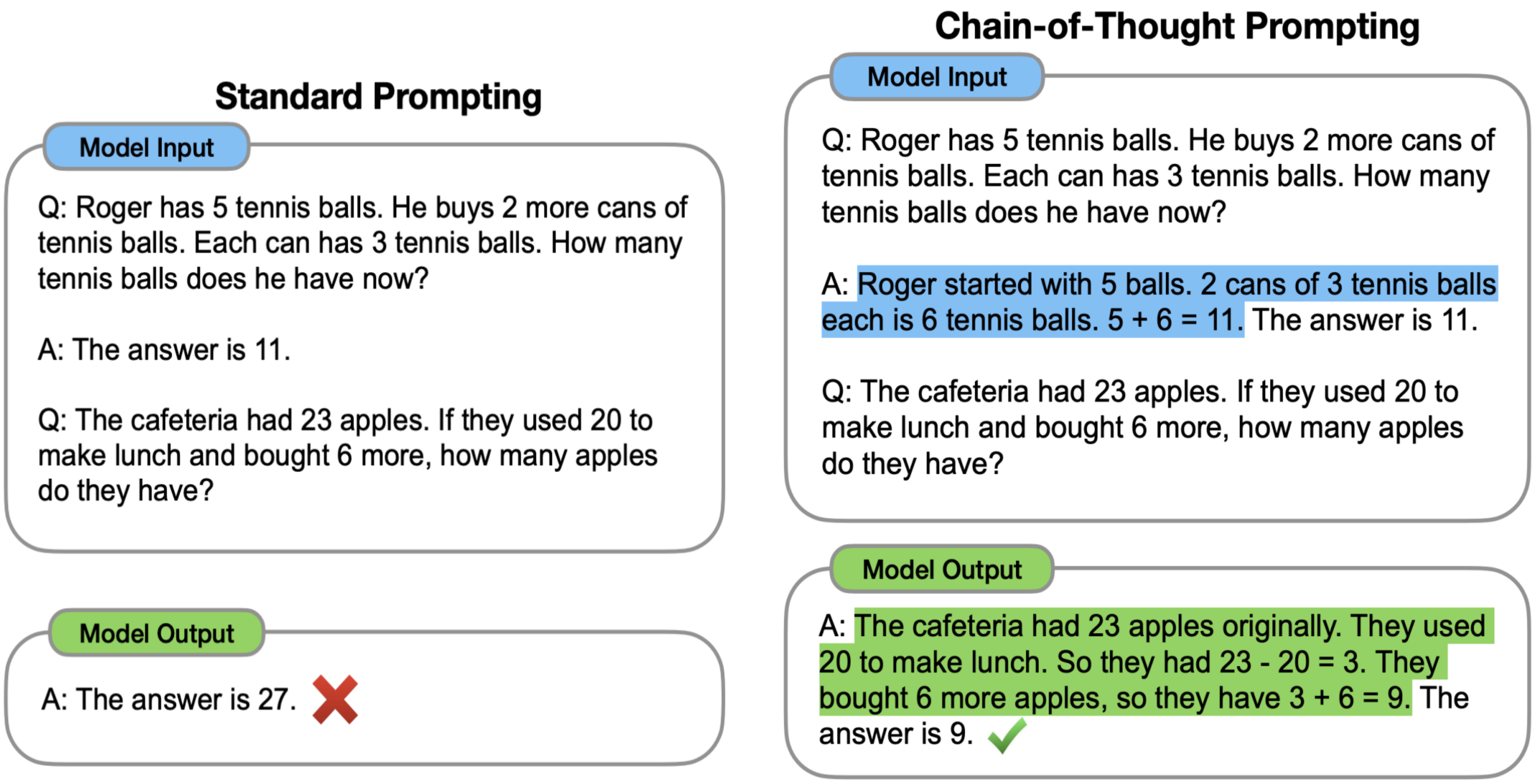
Limitation of Prompts

- 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?

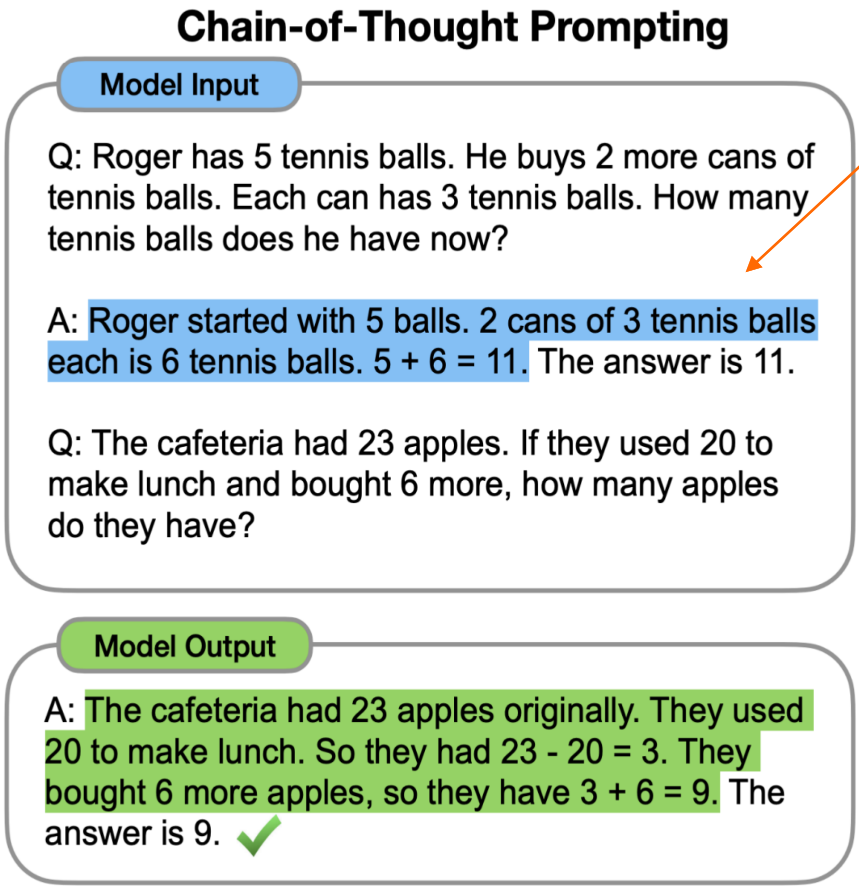
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

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. ✓

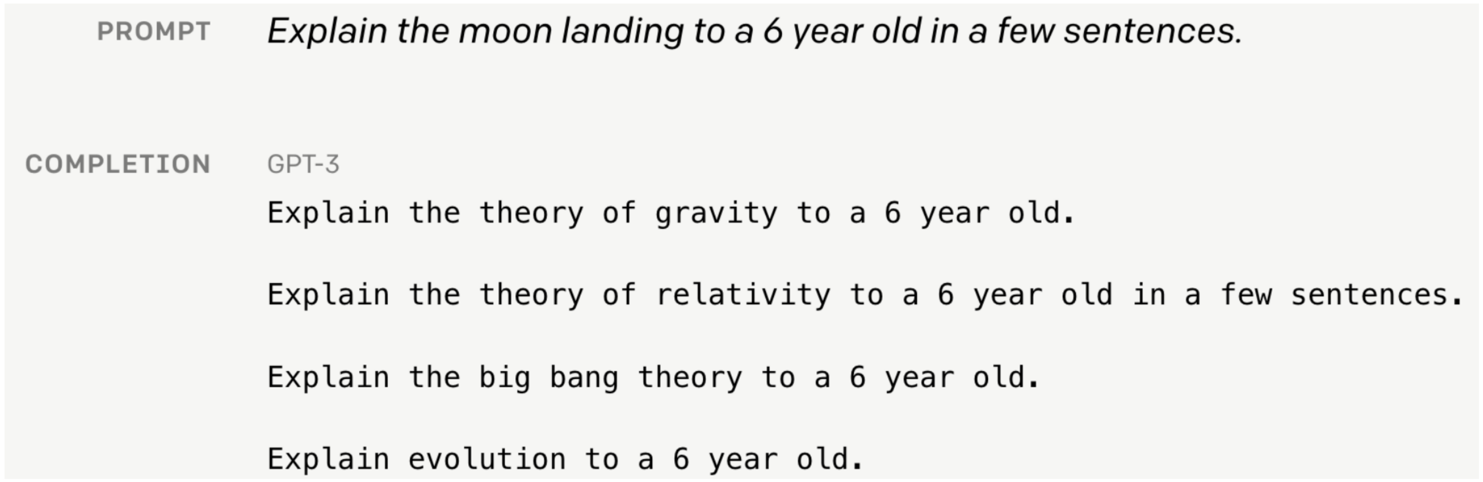
We don't even need to present the reasoning to LLM!

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.*

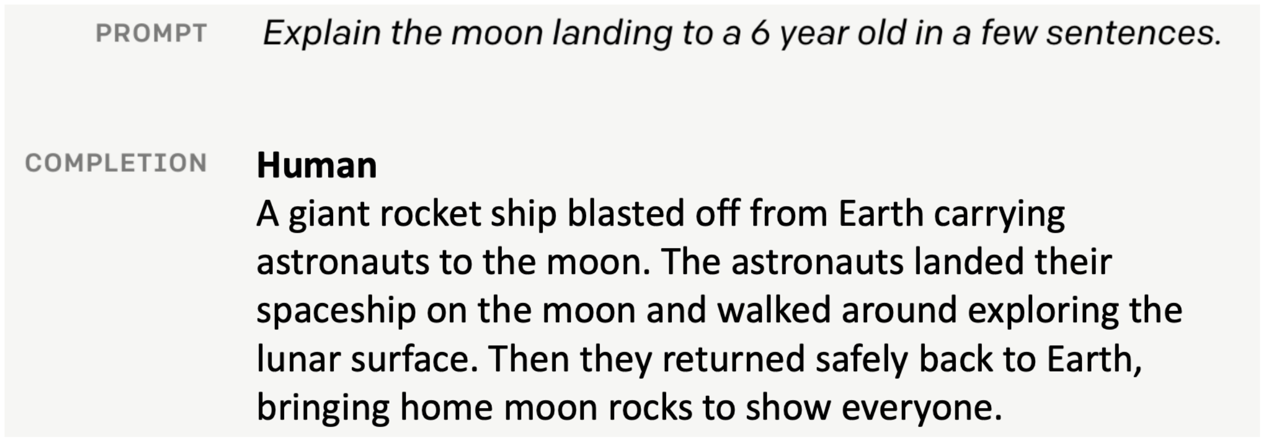
COMPLETION GPT-3

```
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.*

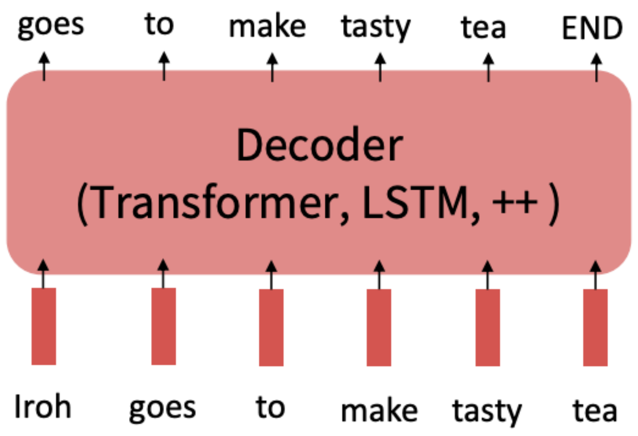
COMPLETION **Human**

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

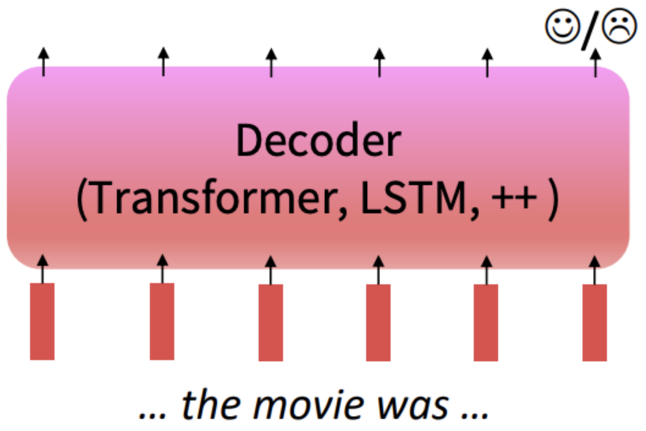
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

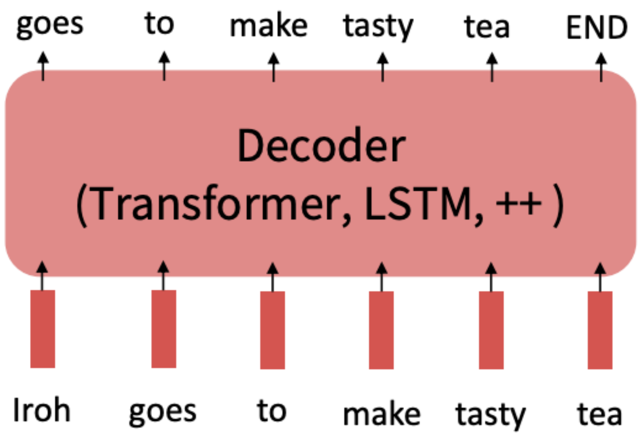
Not many labels; adapt to the task!



Scaling up the finetuning

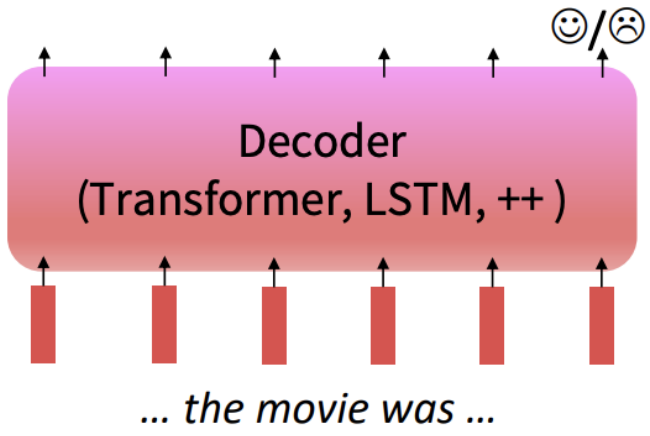
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



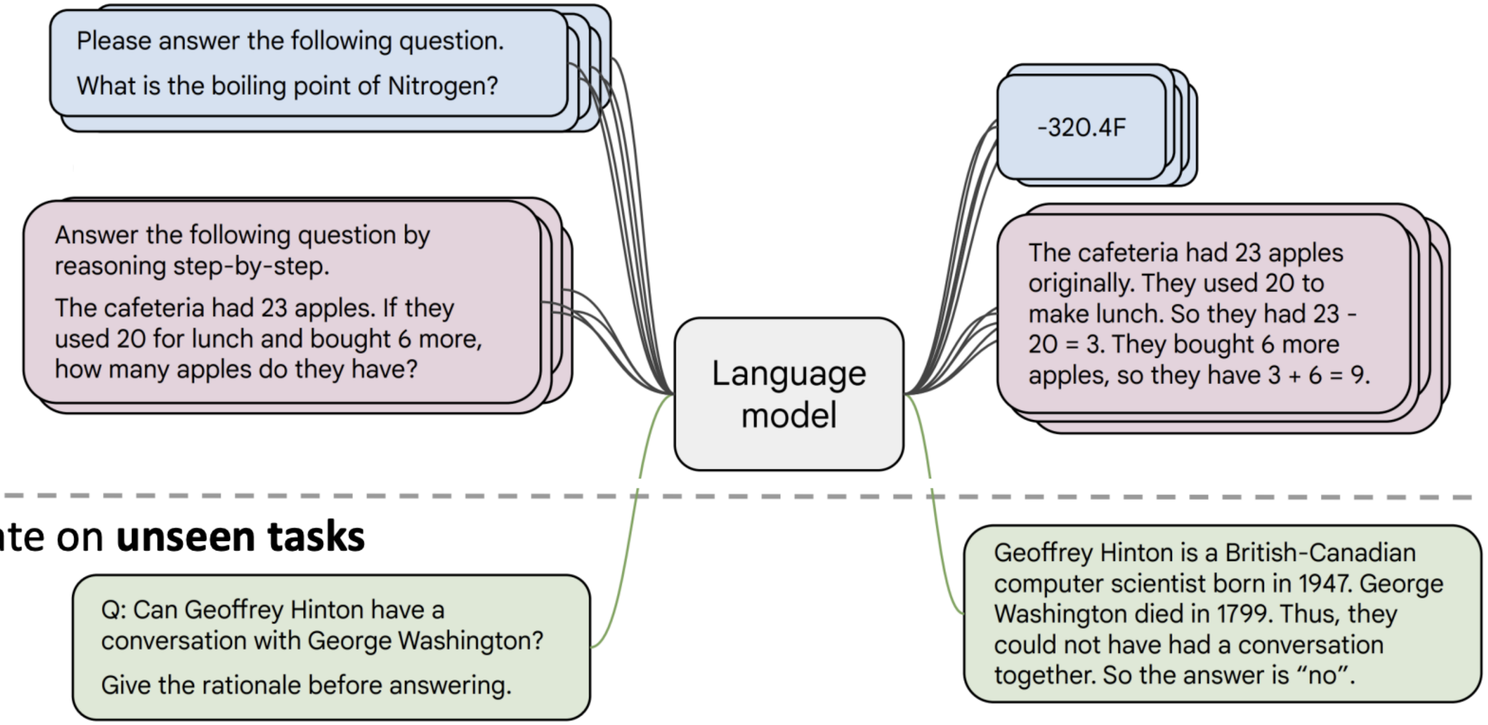
Step 2: Finetune (on many tasks)

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



Instruction Tuning

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


Please answer the following question.
What is the boiling point of Nitrogen?
Answer the following question by reasoning step-by-step.
The cafeteria had 23 apples. If they used 20 for lunch and bought 6 more, how many apples do they have?
Language model
-320.4F
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.
Q: Can Geoffrey Hinton have a conversation with George Washington?
Give the rationale before answering.
Geoffrey Hinton is a British-Canadian computer scientist born in 1947. George Washington died in 1799. Thus, they could not have had a conversation together. So the answer is "no".

- Evaluate on **unseen tasks**