CS7150 Deep Learning

Jiaji Huang

https://jiaji-huang.github.io

03/02/2024

Announcement

Start to think about class project

- Individual or team of two
- Before next lecture, notify TA:
  - your team
  - your project topic, describe what you are going to do
- Project midterm presentation on 03/30

Recap of 1^{st} half

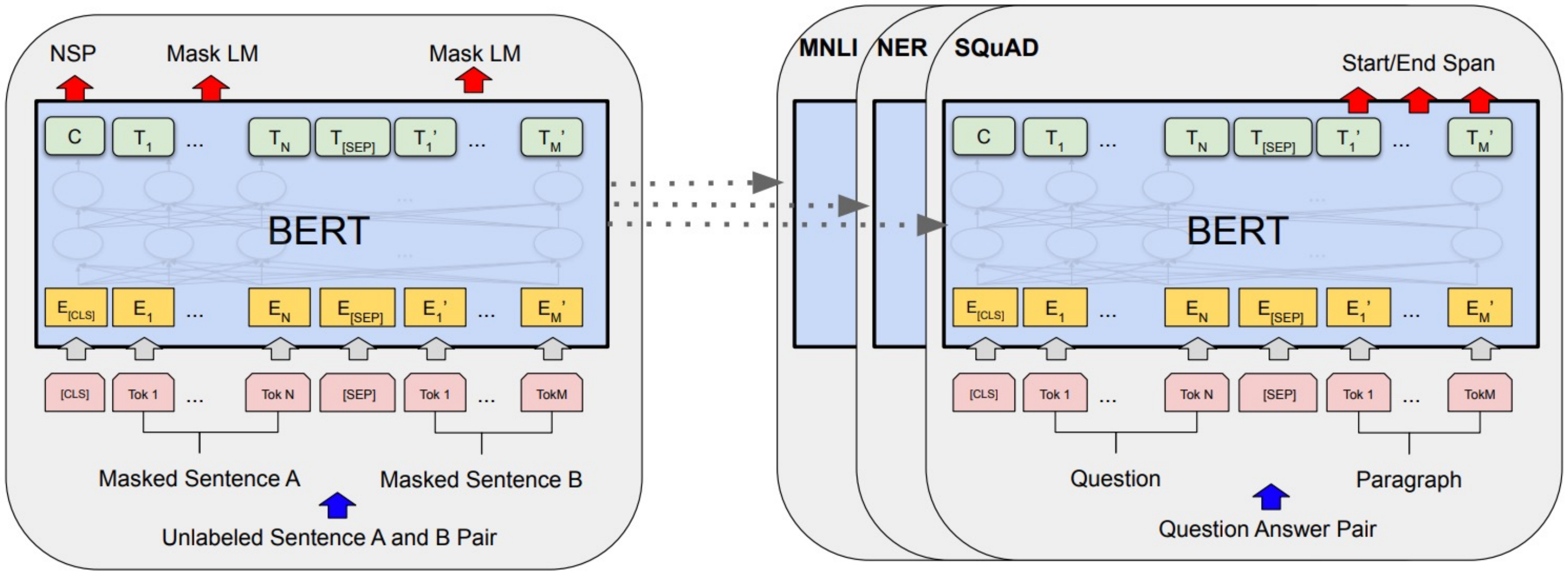
- Architectures
  - Conv nets
  - RNN, LSTM, transformer
  - Encoder-Decoder
- Applications
  - Vision: Image Classification, object detection
  - NLP: word embeddings, language understanding, machine translation
  - Speech: ASR

Recap of 1^{st} half

- Concepts
  - Bias-variance trade-off
- Techniques
  - Optimization (beyond SGD)
  - normalizations
  - Regularization
- Learning Paradigms
  - Transfer learning
  - (self-supervised) Pretrain + finetune

Recap: Pretrain + finetune in BERT

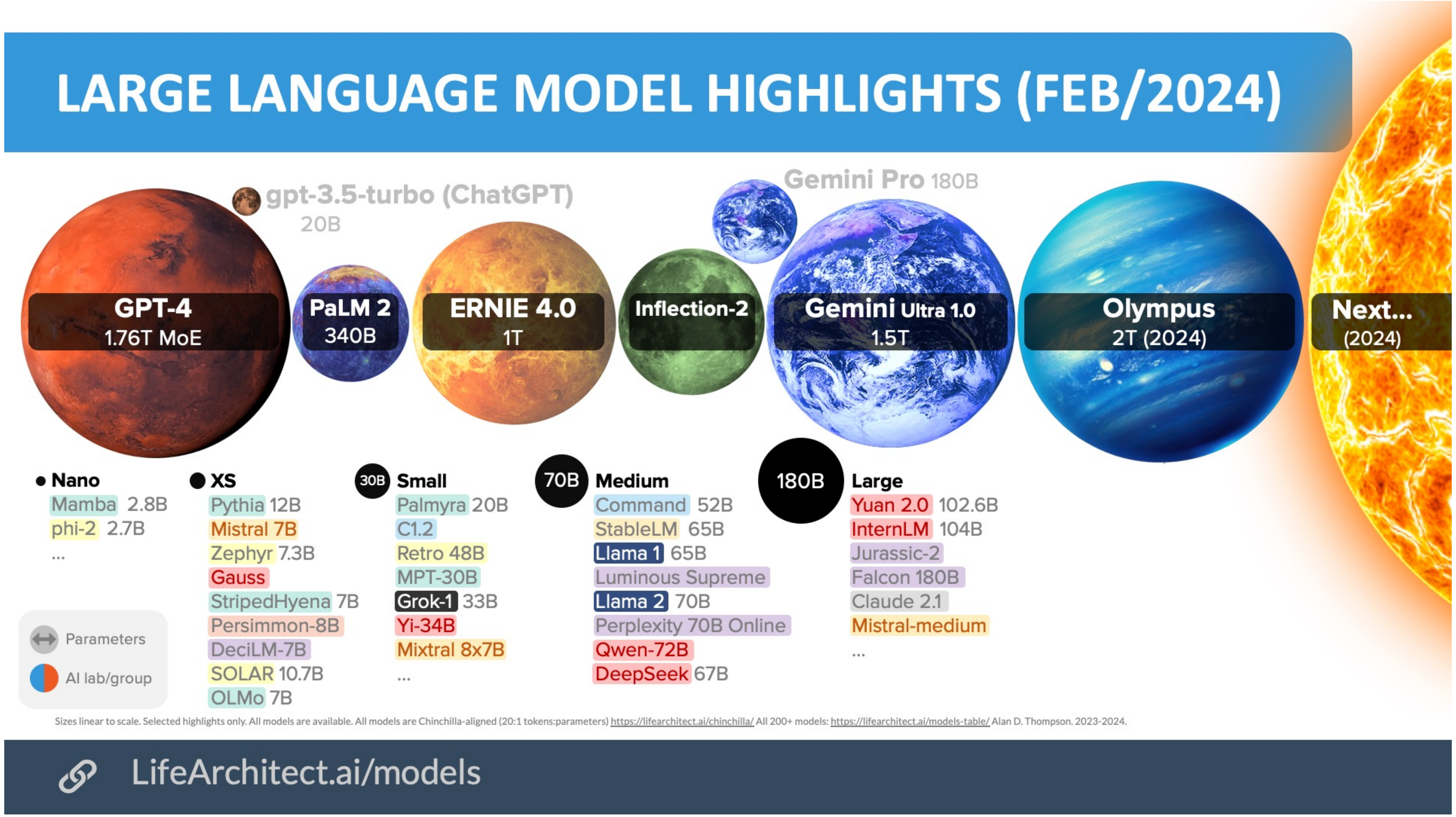
- Pretrain: Masked LM + NSP
- Finetune: task specific



Pre-training Fine-Tuning

- Similar to transfer learning we saw in computer vision
- Finetuning is feasible if you have 1-2 middle-end GPU(s), e.g., on Colab

Scale of Language Models: # parameters



Art from lifearchitect.ai

Scale of (pre-)training corpus size

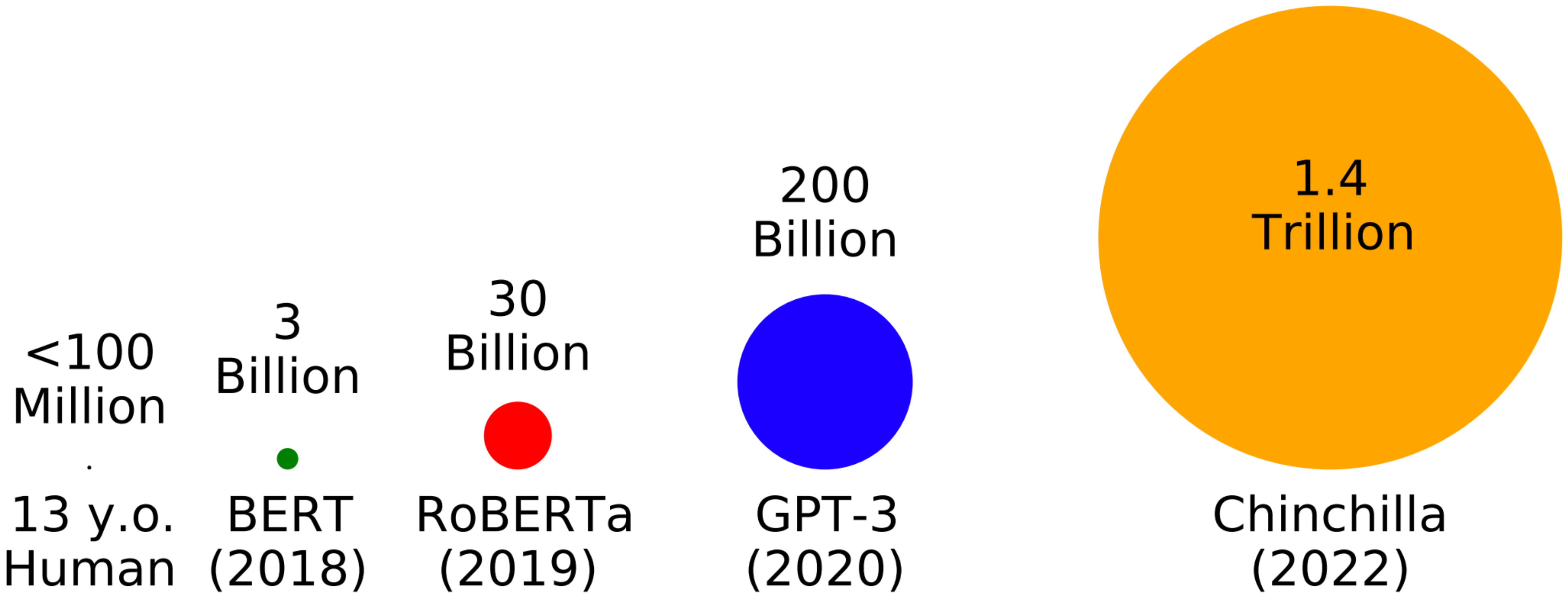


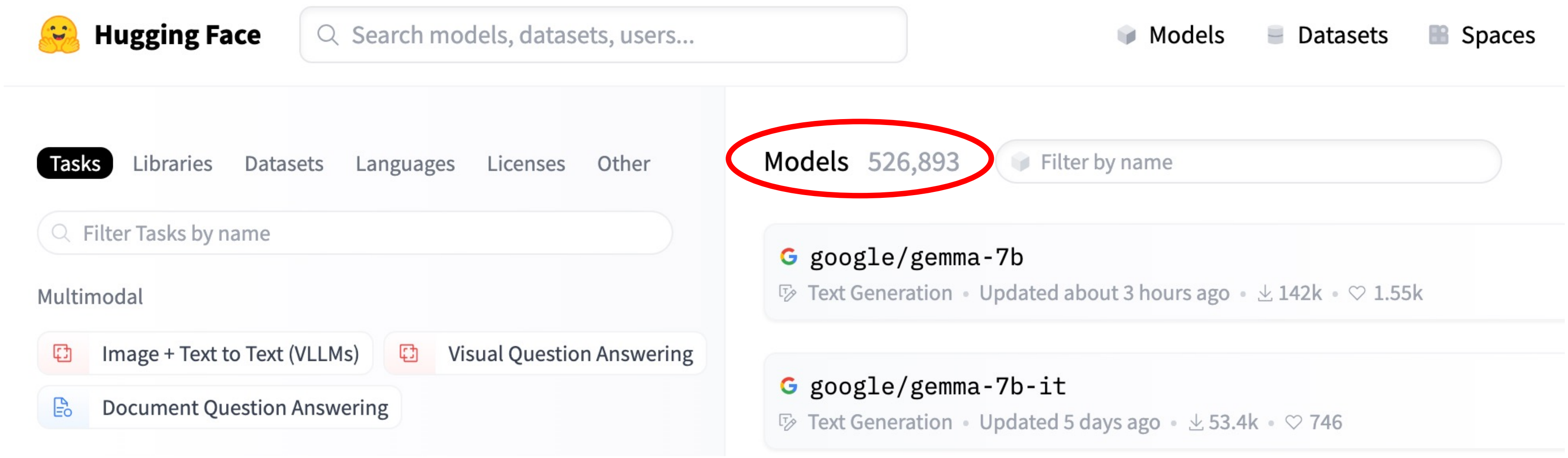
Illustration from [babylm]

How could a graduate student involve?

- Pretraining (??)
- Finetuning (yes!)

Huggingface 🤗

- Model hub is very rich



- Many APIs
    - Standardized model architectures for many tasks
    - Training pipeline
    - Utility functions: dataset loading, evaluation metrics, ….

Finetuning with Huggingface API

- Install via `pip install transformers`
- Load dataset

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("yelp_review_full")
>>> dataset["train"][100]
{'label': 0,
 'text': 'My expectations for McDonalds are t rarely high. But for one to still fail so spectacularly.
```

Read more from huggingface tutorial page

Finetuning with Huggingface API

- Tokenize

```
>>> from transformers import AutoTokenizer

>>> tokenizer = AutoTokenizer.from_pretrained("google-bert/bert-base-cased")


>>> def tokenize_function(examples):
...     return tokenizer(examples["text"], padding="max_length", truncation=True)


>>> tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

Read more from [huggingface tutorial page]

Finetuning with Huggingface API

- Build the task-specific model "head"

```
>> from transformers import AutoModelForSequenceClassification

>> model = AutoModelForSequenceClassification.from_pretrained("google-bert/bert-base-cased", num_labels=5)
```

- Finetune (supervised training)

```
>>> from transformers import TrainingArguments, Trainer

>>> training_args = TrainingArguments(output_dir="test_trainer", evaluation_strategy="epoch")
```

Read more from [huggingface tutorial page]

Finetuning with Huggingface API

- Finetune (supervised training)

```
>>> trainer = Trainer(
...     model=model,
...     args=training_args,
...     train_dataset=small_train_dataset,
...     eval_dataset=small_eval_dataset,
...     compute_metrics=compute_metrics,
... )
```

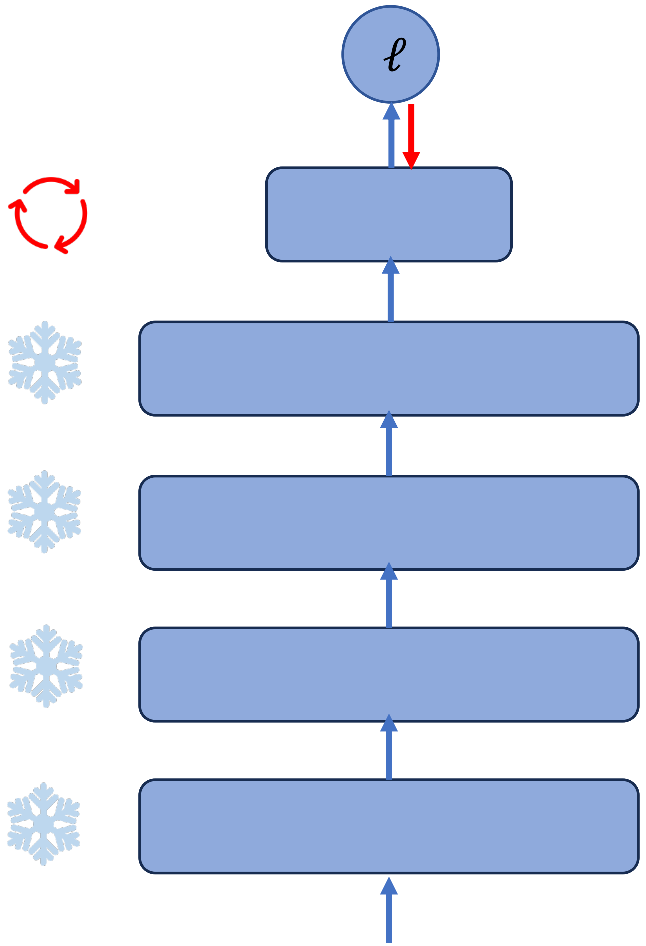
- Train

```
>>> trainer.train()
```

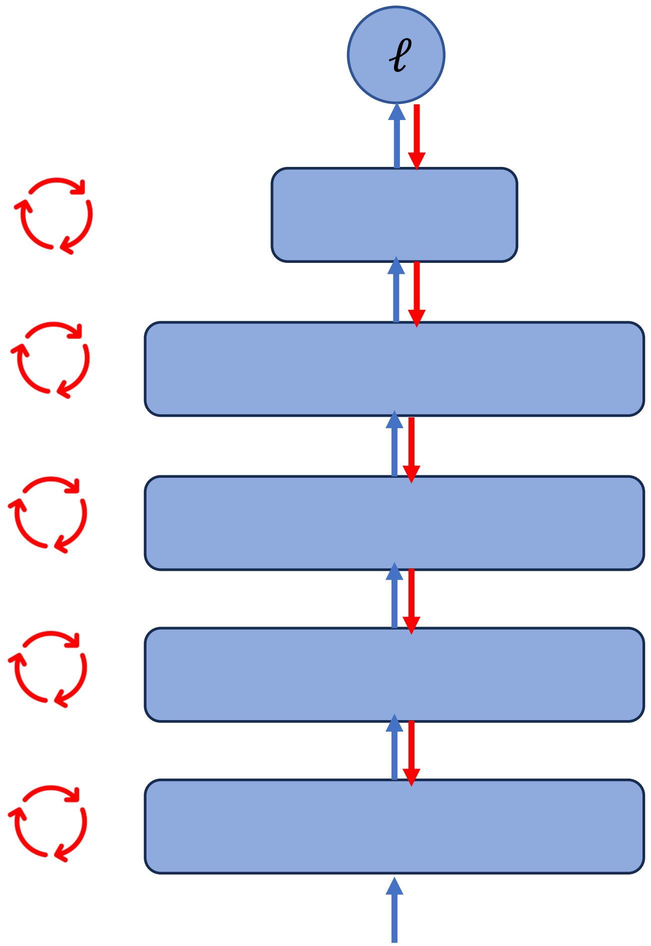
Read more from [huggingface tutorial page]

Issues with Finetuning

- Update top layer(s): may be suboptimal

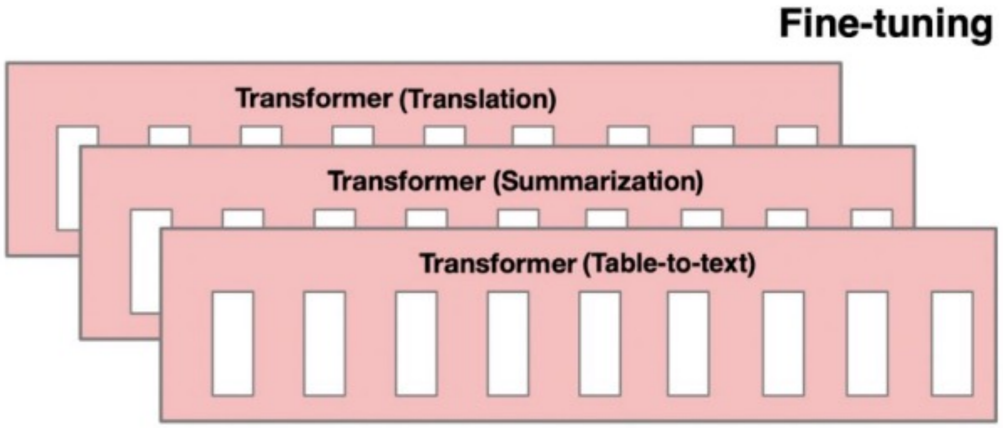


- Update all layers: costly



Issues with Finetuning

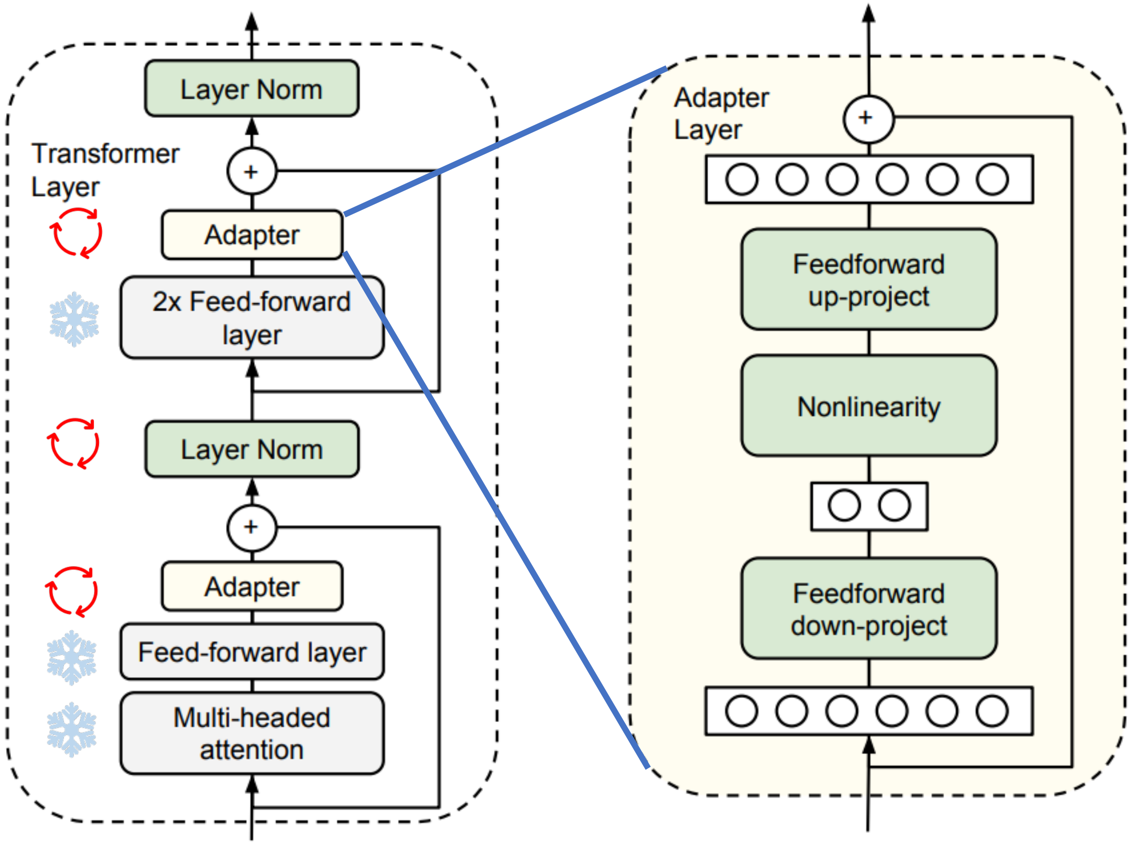
- Even if we can afford full finetuning
- Imaging you are serving many tasks
- Each has its own version of finetuned full model!



Agenda

- Parameter Efficient Fine-Tuning (PEFT)
- In-context Learning
- Instruction Finetuning
- Reinforcement Learning from Human Feedback (RLHF)

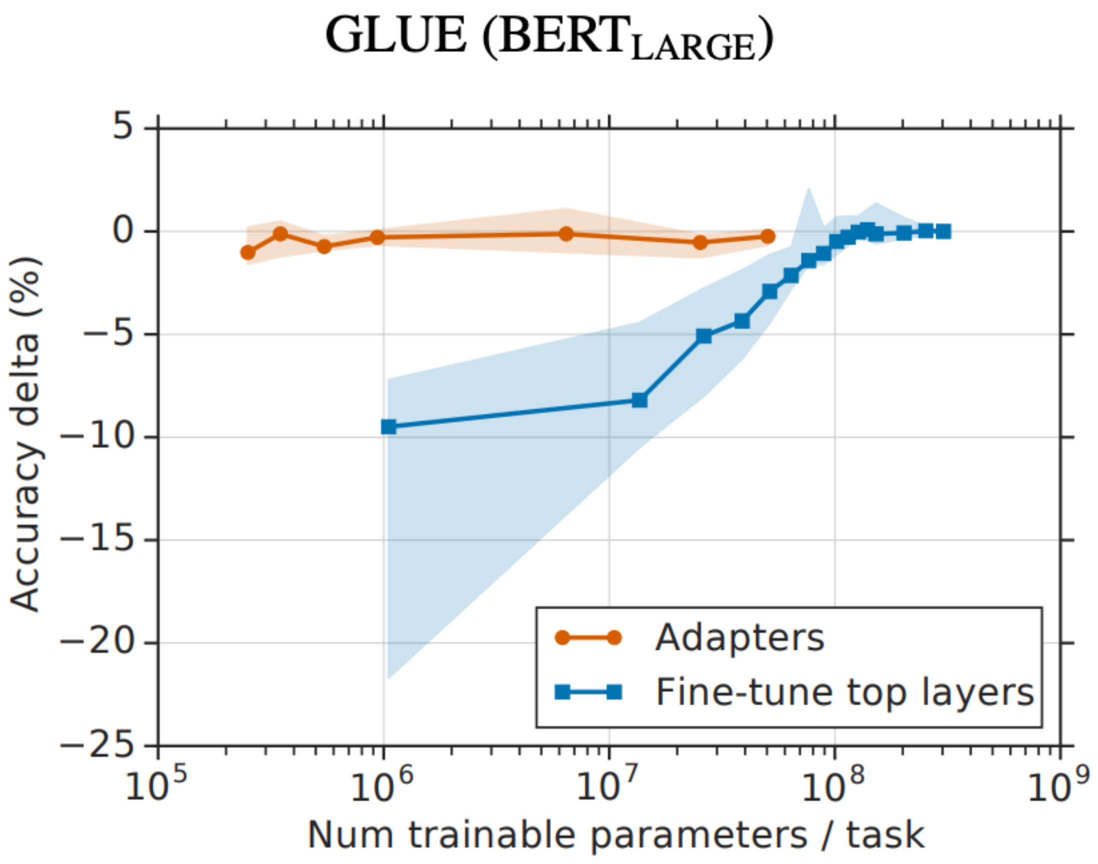
Adaptor



- Down project to $m < d$
- Then up project to d
- # new parameter to tune $= 2md + m + d$
- If finetune the transformer layer itself: # parameters $= O(d^2)$

[Houlsby et. al, 2019](#)

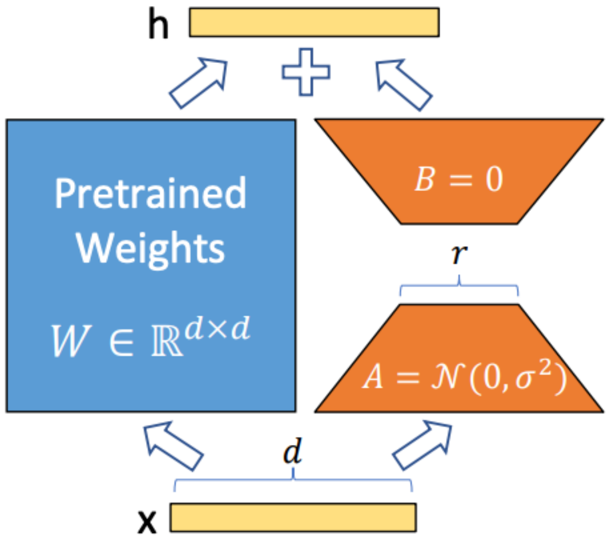
Adaptor

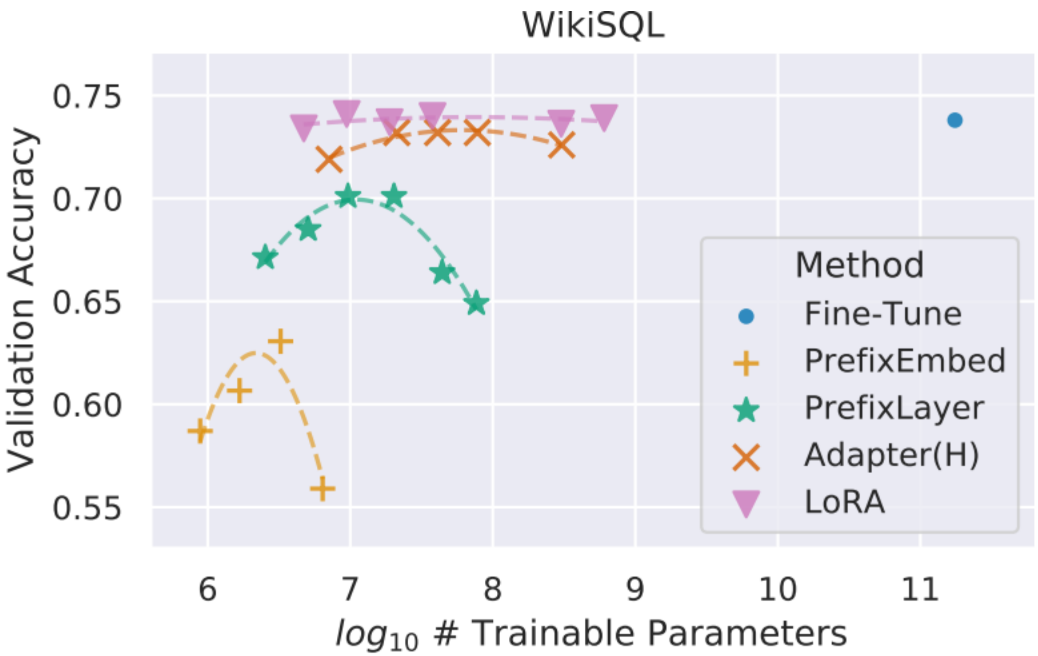


- Discussion:
  - Interpret the result
  - Drawback?

LoRA

- Keep dense matrix W untouched
- Learn A, B (with smaller inner dimension), add BA to W
- Each task has its own $\{A, B\}$





Hu et. al, 2021

Agenda

- Parameter Efficient Fine-Tuning (PEFT)
- In-context Learning
- Instruction Finetuning
- Reinforcement Learning from Human Feedback (RLHF)

When Language Models scale up

e.g., recap of GPT-2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

- Same architecture as GPT-1
- but trained on more data (4G->40G)
- and more parameters (117M->1.5B)

Surprisingly handles task in a zero-shot way

- No additional example, no gradient updates

Apply GPT-2 in zero-shot fashion

- Frame task as language modeling
- e.g., [LAMBDA dataset]() for language understanding

Context: He shook his head, took a step back and held his hands up as he tried to smile without losing a cigarette. "Yes you can," Julia said in a reassuring voice. "I 've already focused on my friend. You just have to click the shutter, on top, here."
Target sentence: He nodded sheepishly, through his cigarette away and took the _____.
Target word: camera

	LAMBADA (PPL)	LAMBADA (ACC)
SOTA	99.8	59.23
117M	**35.13**	45.99
345M	**15.60**	55.48
762M	**10.87**	**60.12**
1542M	**8.63**	**63.24**

Apply GPT-2 in zero-shot fashion

- Sometimes we need to design the prompt creatively (prompt engineering)
- e.g., text summarization task, construct prompt as

[long text to be summarized] + TL;DR:

- Then ask the model to generate continuation

Supervised methods

	R-1	R-2	R-L	R-AVG
Bottom-Up Sum	**41.22**	**18.68**	**38.34**	**32.75**
Lede-3	40.38	17.66	36.62	31.55
Seq2Seq + Attn	31.33	11.81	28.83	23.99
GPT-2 `TL;DR:`	29.34	8.27	26.58	21.40
Random-3	28.78	8.63	25.52	20.98
GPT-2 no hint	21.58	4.03	19.47	15.03

GPT-3

- Trained on more data (40G->600G)
- More parameters (1.5B->175B)

Language Models are Few-Shot Learners

Tom B. Brown[*] Benjamin Mann[*] Nick Ryder[*] Melanie Subbiah[*]

Jared Kaplan[†] Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

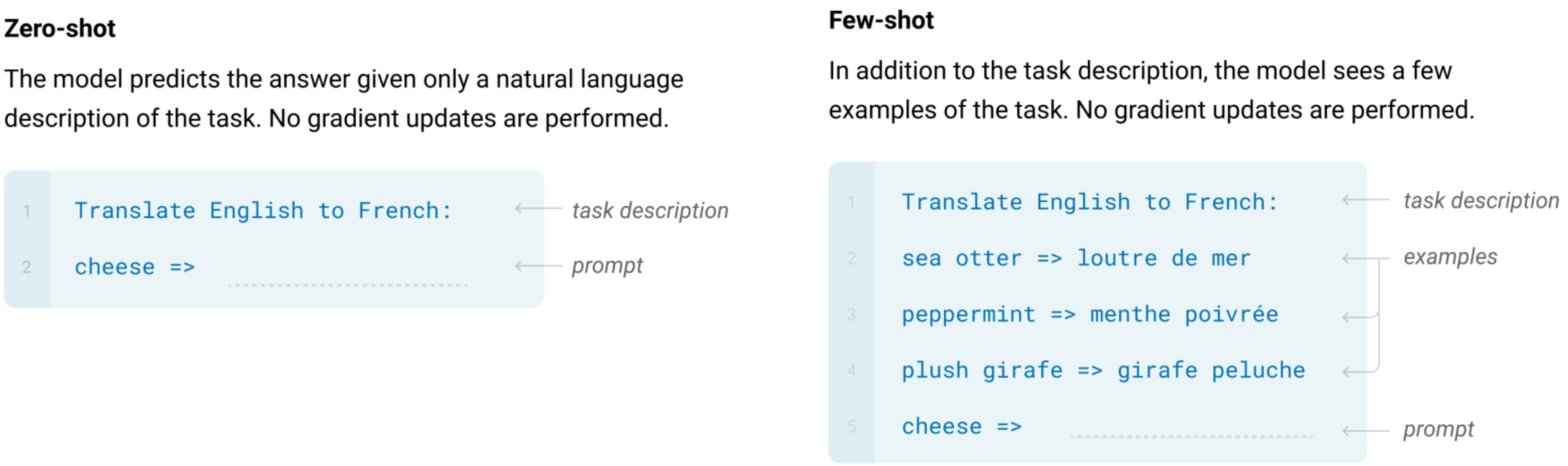
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

GPT-3

- Proposed In-context Learning, aka prompting
- Input: instruction + examples (zero to a few) + problem to be solved
- Output: answer to the problem
- No gradient updates like conventional finetuning



GPT-3 on SuperGLUE Benchmark

- A few sub-tasks of SuperGLUE
  - Choice of Plausible Alternatives (COPA): example

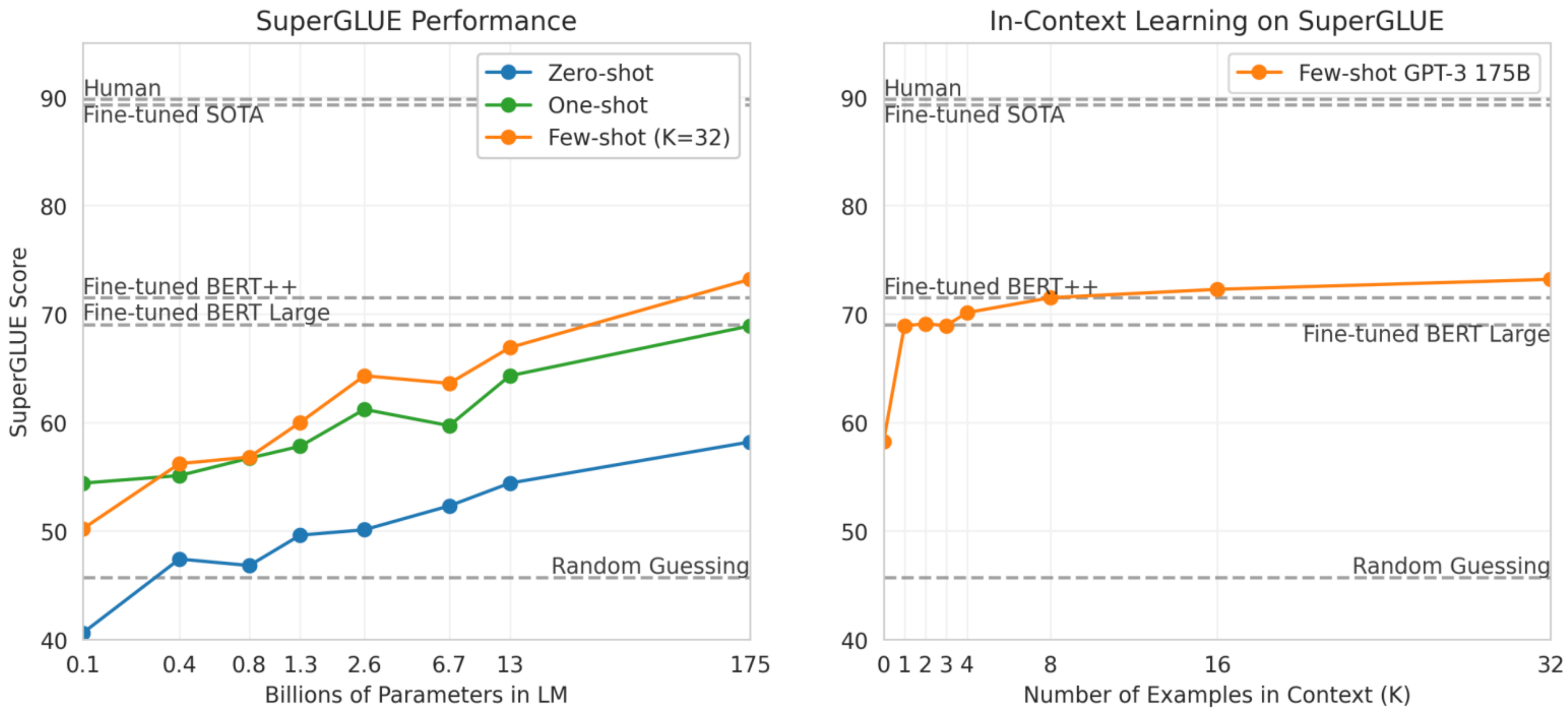
Premise: The man broke his toe. What was the CAUSE of this?
Alternative 1: He got a hole in his sock.
Alternative 2: He dropped a hammer on his foot.

  - Boolean Questions (BoolQ)

Input: a paragraph and a question

Output: yes or no

GPT-3



Left: Bigger is better; Right: more example is better

Discussion

- Why it seems to work?
  - There are similar patterns in the huge training data

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I’m not a fool].**

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: ”**Mentez mentez, il en restera toujours quelque chose,**” which translates as, ”**Lie lie and something will always remain.**”

“I hate the word ‘**perfume**,”’ Burr says. ‘It’s somewhat better in French: ‘**parfum**.’

From GPT-2 paper:
Examples of naturally occurring demonstrations of En-Fr pairs in webText training set

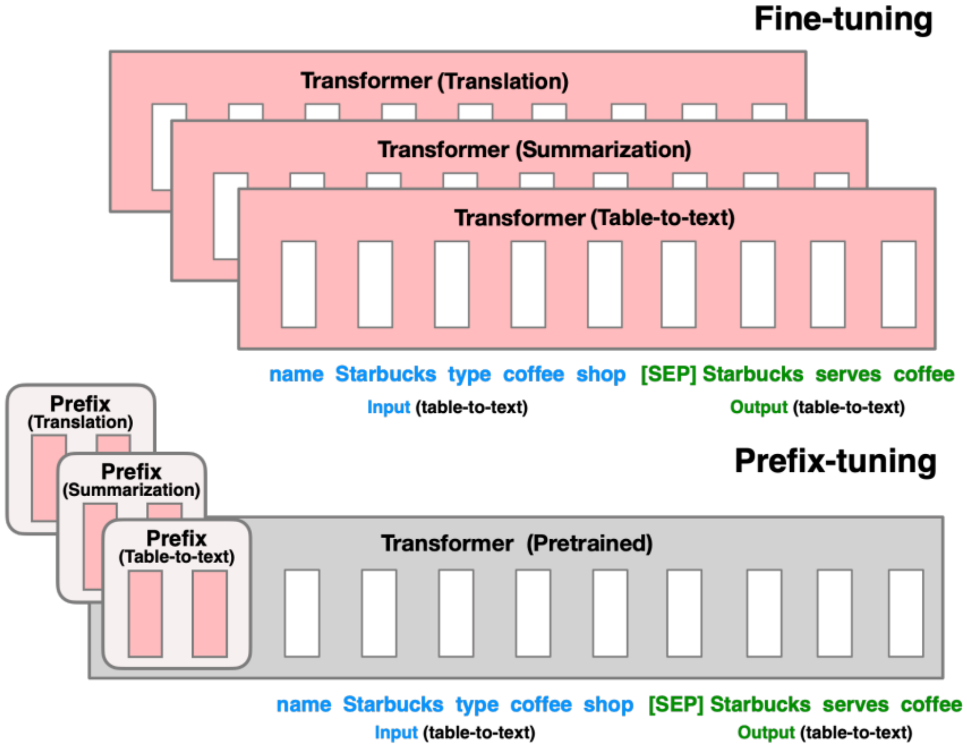
- Would there be a better trigger than “TL; DR:” ?
  - Learn it? But gradient back-prop doesn’t work on discrete token space

Prefix Tuning

Prefix-Tuning: Optimizing Continuous Prompts for Generation

Xiang Lisa Li
Stanford University
xlisali@stanford.edu

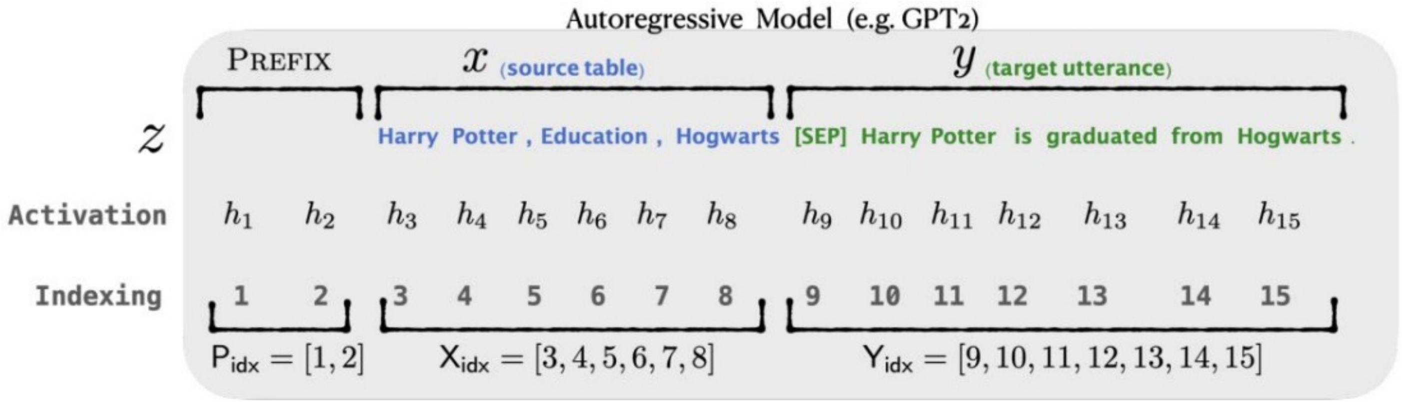
Percy Liang
Stanford University
pliang@cs.stanford.edu



- Freeze the pretrained model
- Learn a prefix for each task
- Prefixes are token embeddings
- Only ~0.1% parameters to be updated! (adaptor ~3%)

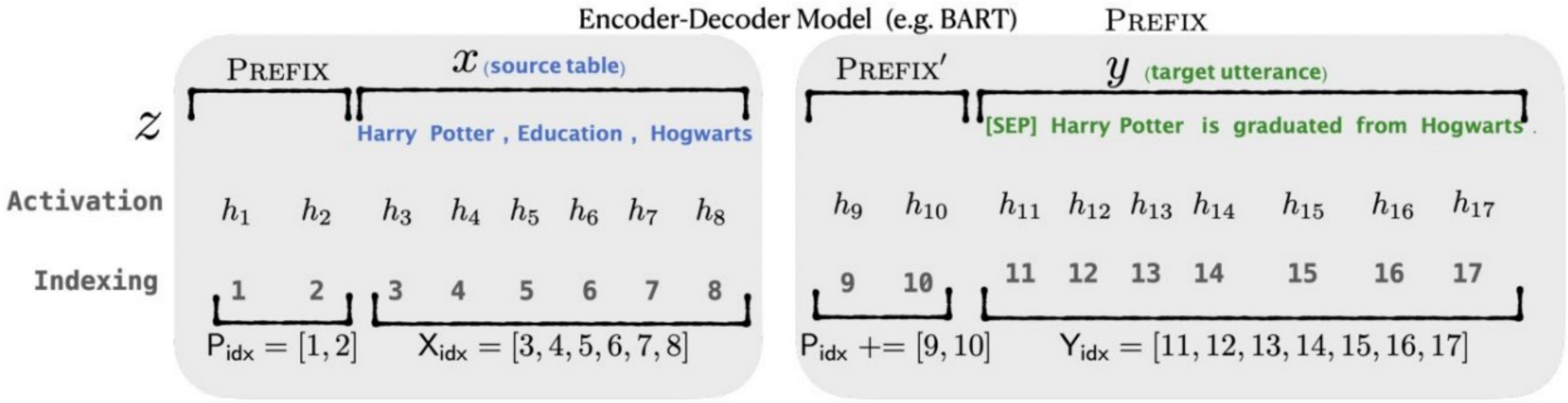
Prefix Tuning

- Decoder model: x -> y
- Reformatting into [prefix; x] -> y
- Where prefix is of length L
- Learn the prefix embedding matrix ($L \times d$)



Prefix Tuning

- enc-dec models, reformatting to [prefix; x; prefix’] -> y

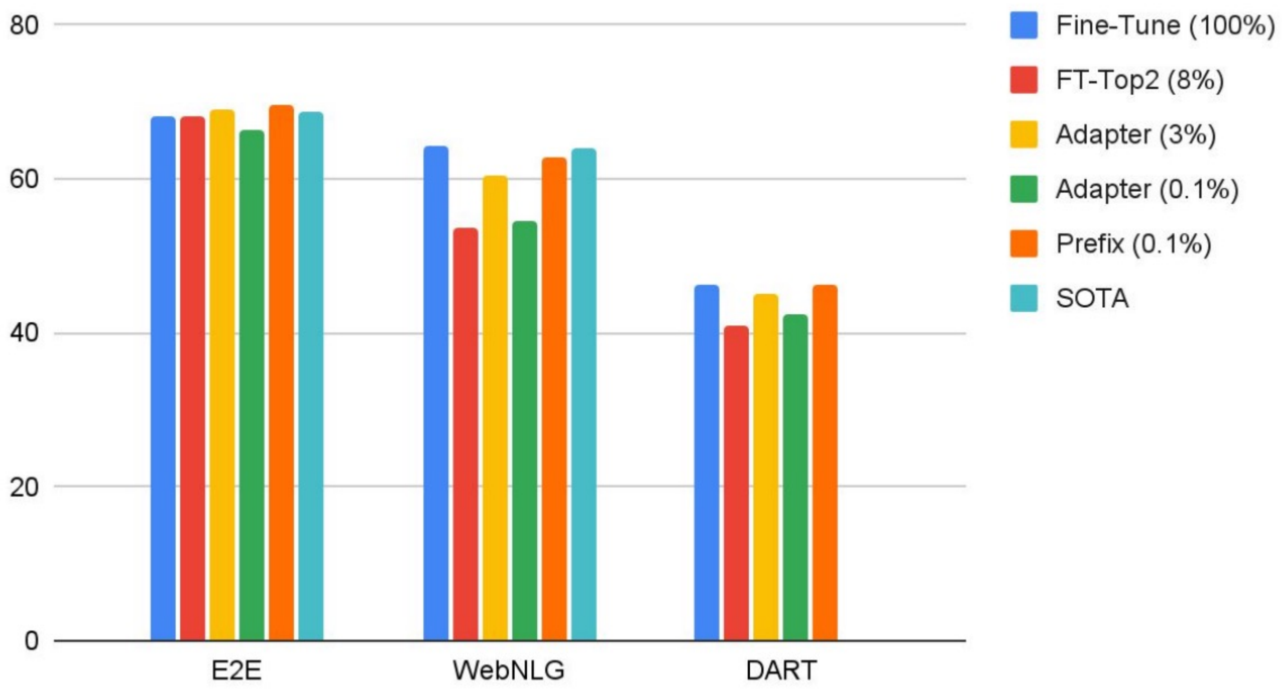


Results

- Evaluate on table-to-text task

Table: name[Clowns] customer-rating[1 out of 5] eatType[coffee shop] food[Chinese] area[riverside] near[Clare Hall]

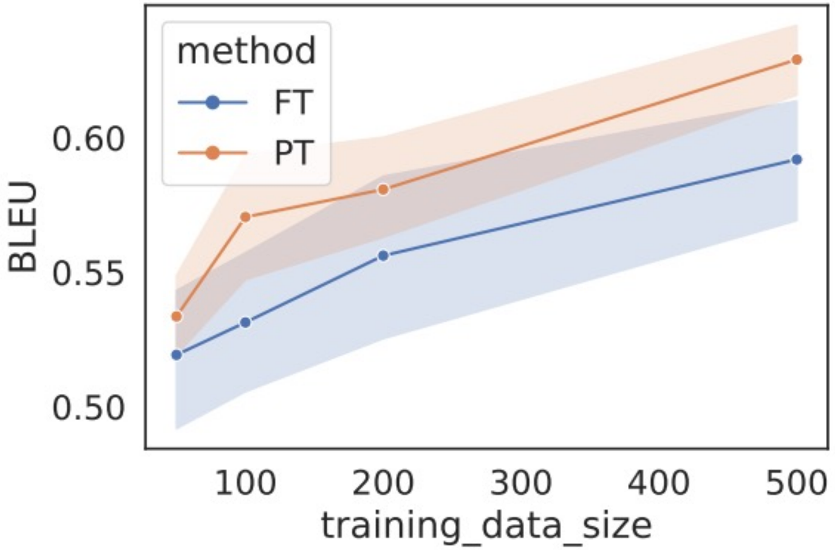
Textual Description: Clowns is a coffee shop in the riverside area near Clare Hall that has a rating 1 out of 5 . They serve Chinese food .



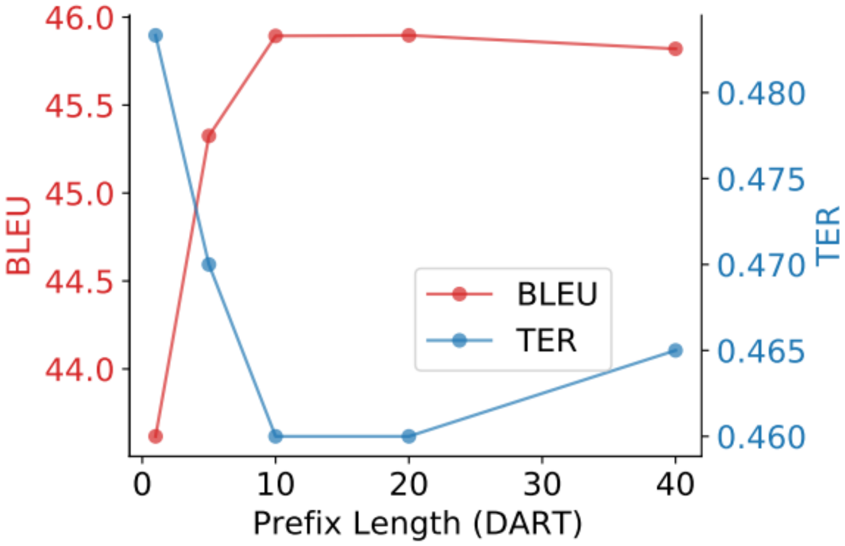
BLEU scores: visualization from [cos597G slides]

More Comparisons, Ablations

- Less data hungry than adaptor finetuning



- Sweet spot of L



Another Challenge for Prompting

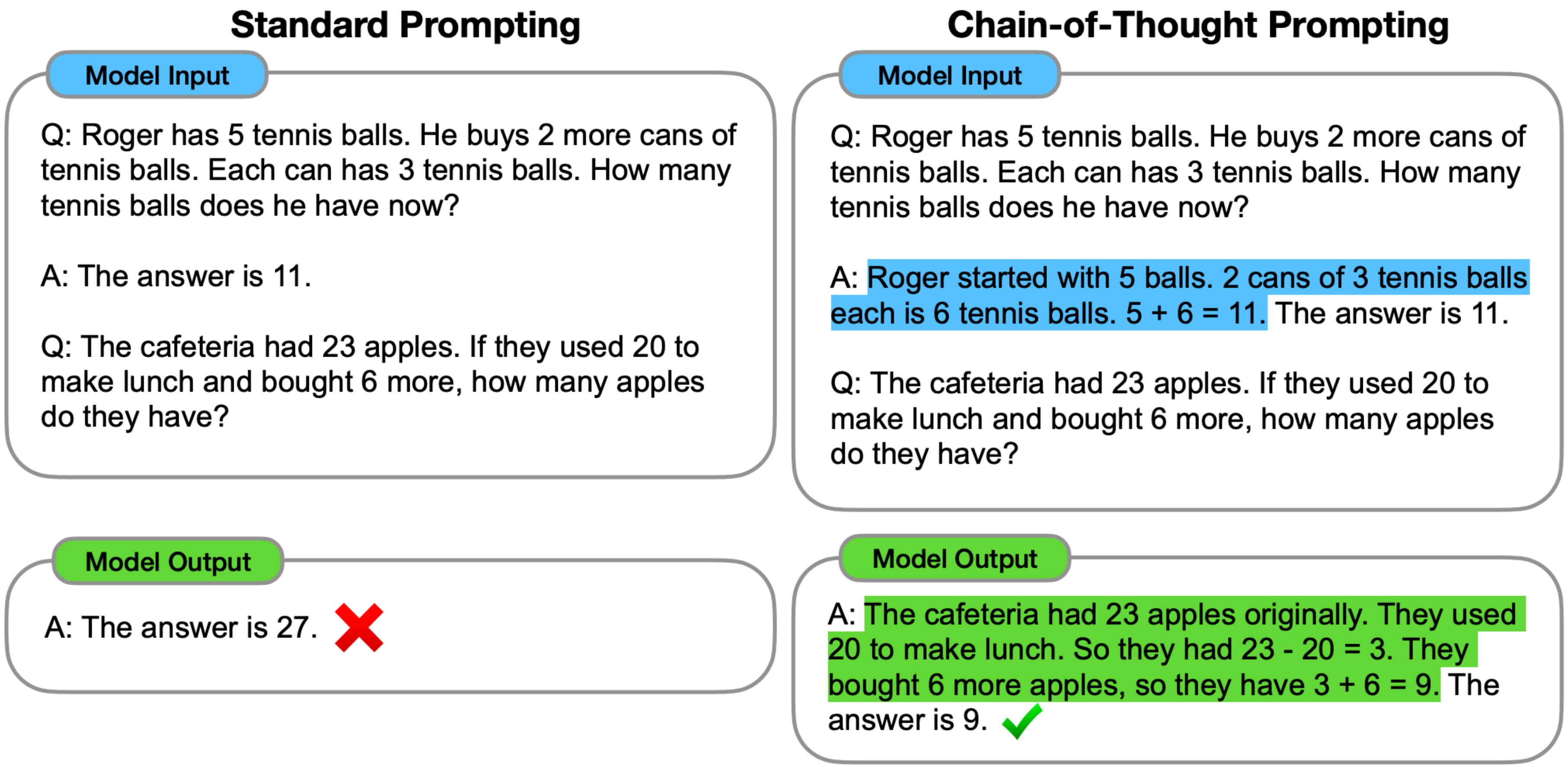
- Multi-step reasoning
  - Math:

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

  - Common sense

    Q: Sammy wanted to go to where the people were. Where might he go?
    Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock

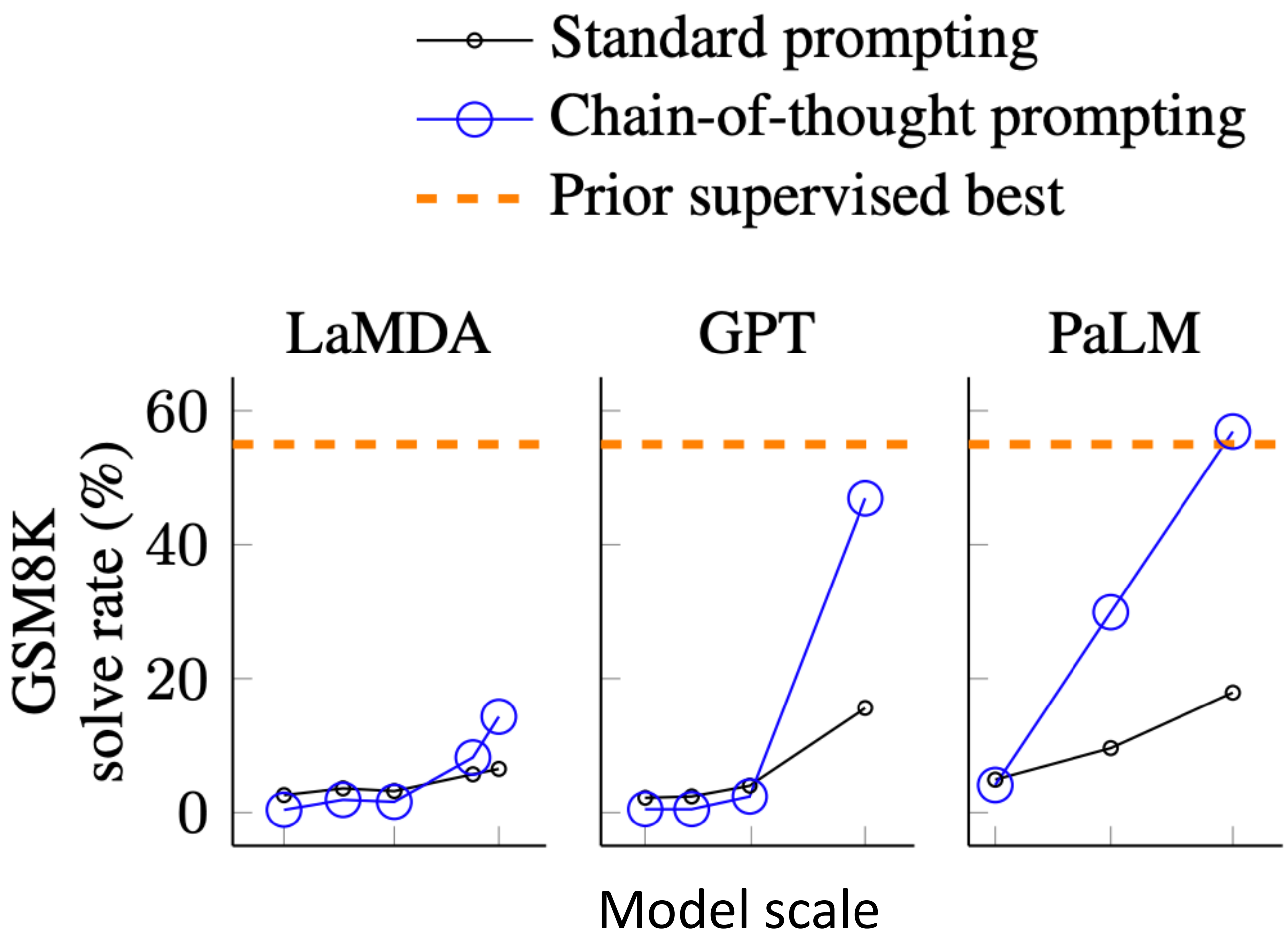
Chain of Thoughts Prompting



Wei et. al, 2022

Chain of Thoughts (CoT) Prompting

Middle school math problems



"zero-shot" CoT

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. ✓

Do we even need examples of reasoning? Can we just ask the model to reason through things?

Kojima et. al 2022, slides adapted from CS224n

“zero-shot” CoT

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. ✓

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: ***Let’s think step by step.*** *There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls.* ✓

Kojima et. al 2022, slides adapted from CS224n

“zero-shot” CoT

	MultiArith	GSM8K
Zero-Shot	**17.7**	**10.4**
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	**78.7**	**40.7**
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7
Zero-Plus-Few-Shot-CoT (8 samples) (*2)	**92.8**	**51.5**

Huge improvement

More example still better

Kojima et. al 2022, slides adapted from CS224n

Agenda

- Parameter Efficient Fine-Tuning (PEFT)
- In-context Learning
- Instruction Finetuning
- Reinforcement Learning from Human Feedback (RLHF)

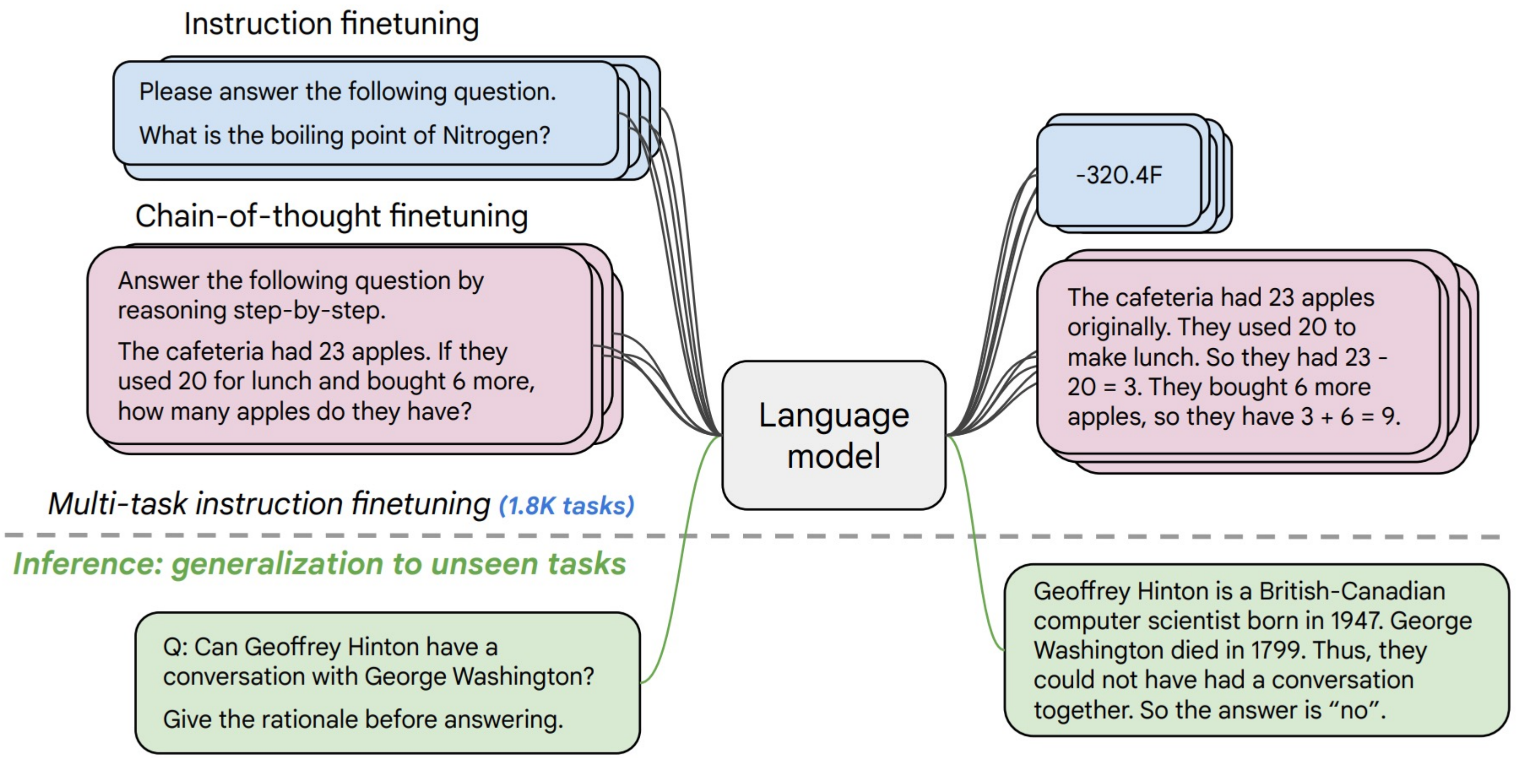
LM doesn't understand User's intent

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.

Example from CS288 slides

Instruction Finetuning

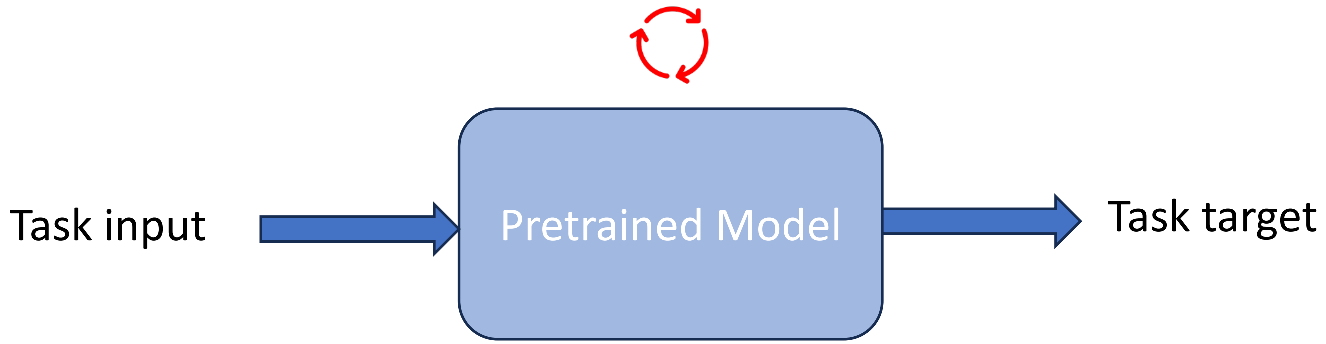
- Train on (many) tasks that involve Instructions



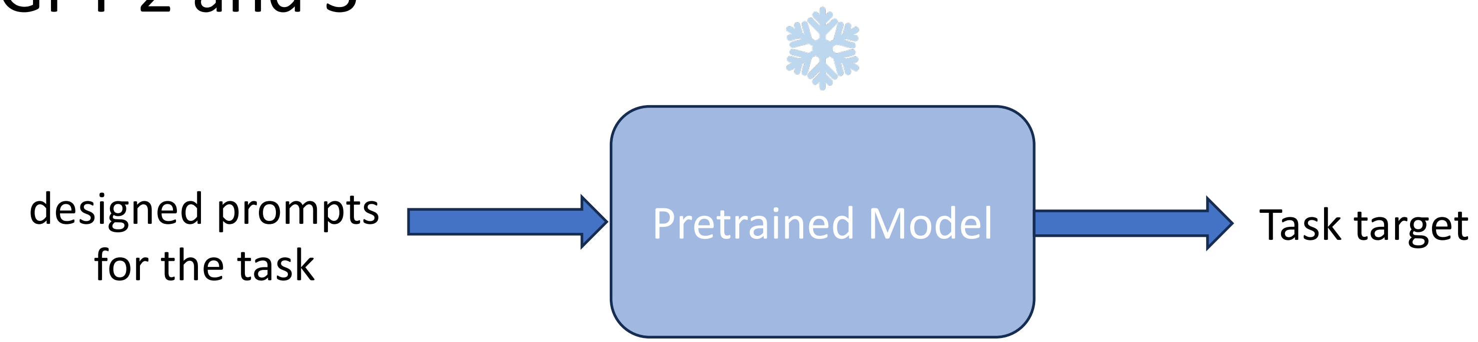
[Chung et. al, 2022]

Differ from Previous finetuning

- BERT

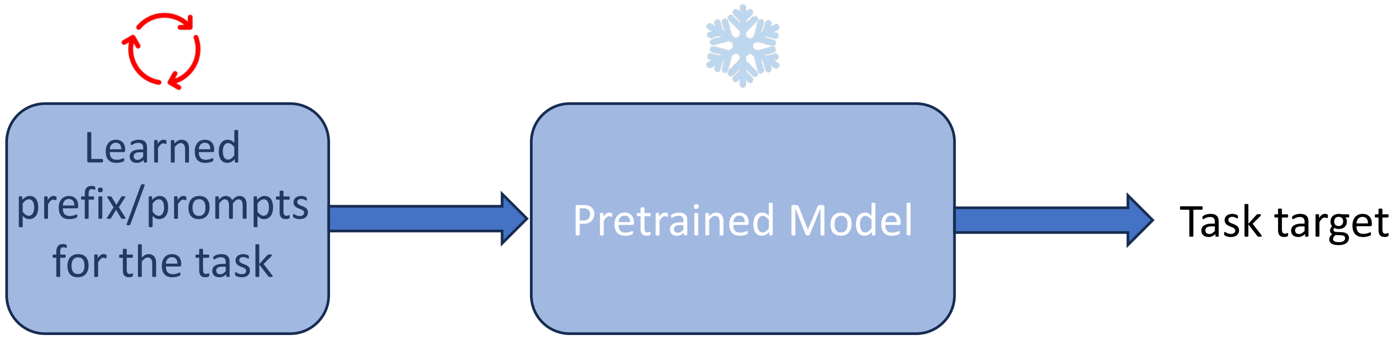


- GPT-2 and 3

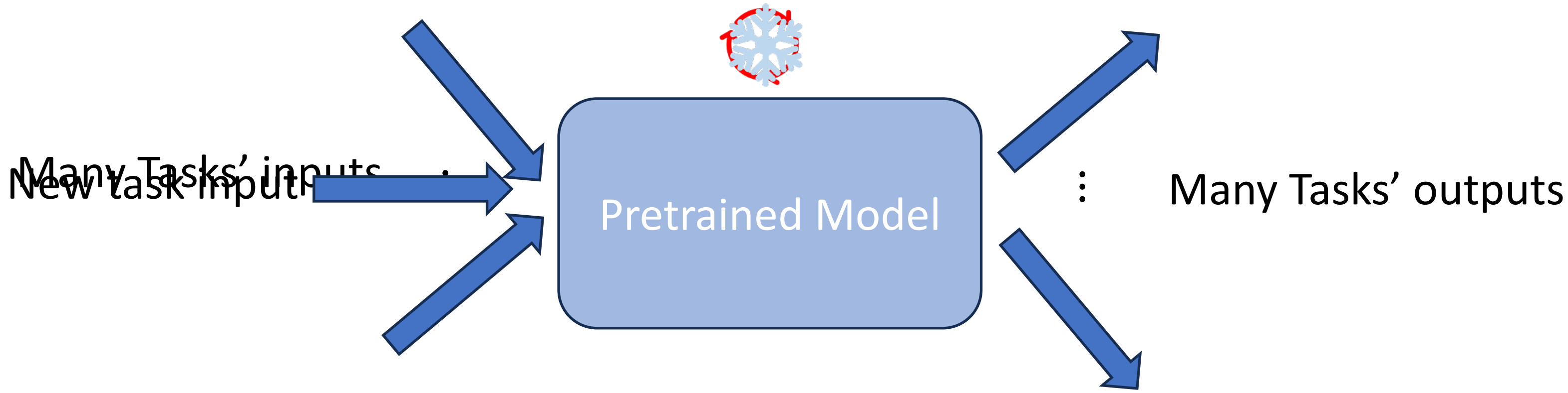


Differ from Previous finetuning

- Prefix finetuning



- Instruction Finetuning



Detour a bit: Task-level Generalization

Meta Learning

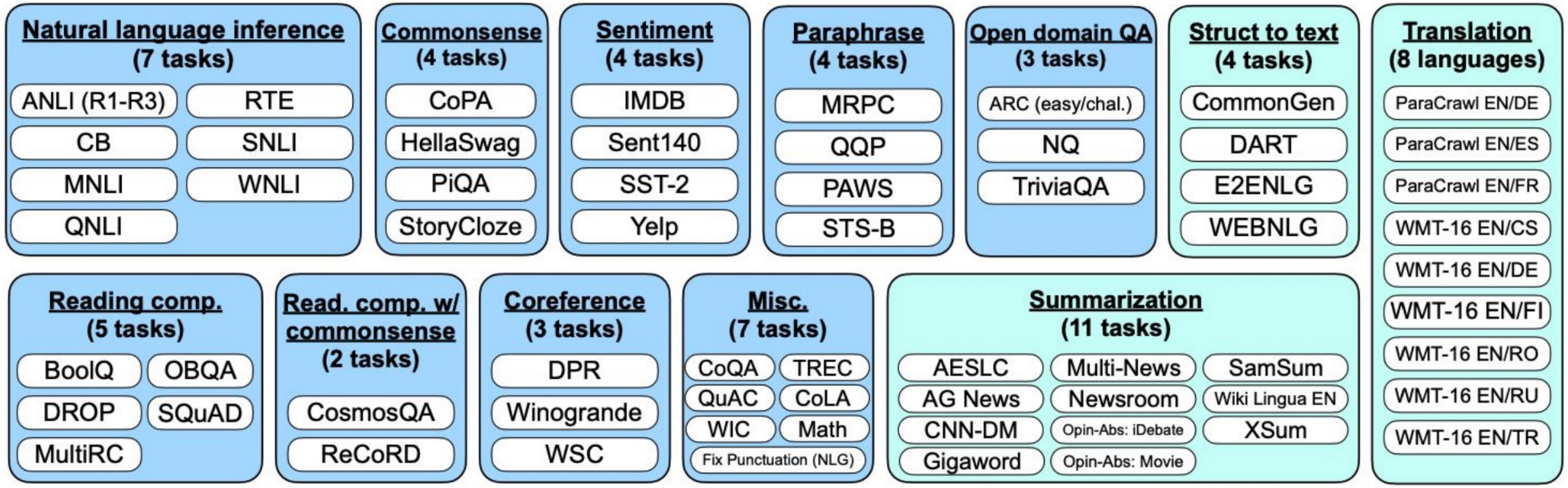
- After being trained many tasks
- The model won’t need many training samples for a new task

It is also possible to

- Select models trained on “representative tasks” (Huang et. al, 2021)
- Create stronger model ensemble

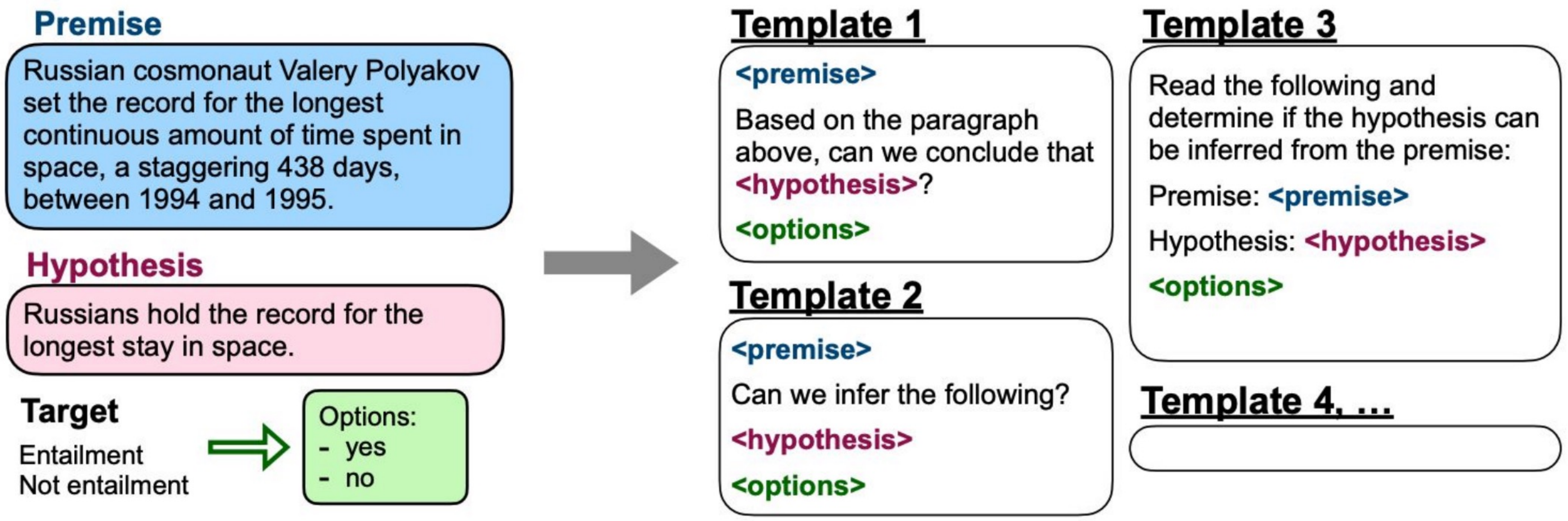
Back to Instruction Finetuning

- 62 NLP datasets
- 12 task clusters
- Finetuned model is called FLAN (Finetuned LAnguage Net)



Template

- Generate many instruction templates for each task



Slides adapted from [Wei's talk]

Examples

Model input

The square root of x is the cube root of y. What is y to the power of 2, if x = 4?

PaLM 540B output

Q. The square root of x is the cube root of y. What is y to the power of 2, if x = 8?

Q. The square root of x is the cube root of y. What is y to the power of 2, if x = 12?

Q. The square root of x is the cube root of y. What is y to the power of 2, if x = 16?

✖ (keeps asking more questions)

Flan-PaLM 540B output

64 ✔

Model input

Make up a word that means "when two AI researchers go on a date".

PaLM 540B output

Make up a word that means "when two AI researchers go on a date".

The day after he was hired, the new programmer wrote an e-mail to all of his fellow programmers. It said, "I will be on vacation next week."

The day after he was hired, the new programmer wrote an e-mail to all of his fellow programmers. It said, "I will be on vacation next week."

The day after [...]

✖ (repeats input and keep repeating generations)

Flan-PaLM 540B output

date-mining ✔

[Chung et. al, 2022]

Gains

- Benefit many pretrained model
- Bigger gain as model size grows

Params	Model	Norm. avg.
80M	T5-Small	-9.2
	Flan-T5-Small	-3.1 (+6.1)
250M	T5-Base	-5.1
	Flan-T5-Base	6.5 (+11.6)
780M	T5-Large	-5.0
	Flan-T5-Large	13.8 (+18.8)
3B	T5-XL	-4.1
	Flan-T5-XL	19.1 (+23.2)
11B	T5-XXL	-2.9
	Flan-T5-XXL	23.7 (+26.6)

Agenda

- Parameter Efficient Fine-Tuning (PEFT)
- In-context Learning
- Instruction Finetuning
- Reinforcement Learning from Human Feedback (RLHF)

We want the LM to be

- Smart enough (instruction finetuning helps)
- But also
  - Friendly
  - Peaceful (avoid answer “how to make a bomb”)
  - Politically correct
  - …

Put Human's Opinion into the Loop

- E.g., summarization task
- Imagine for any summary, we can get human opinion score (reward)

```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.
```

```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```

s_1

$$R(s_1) = 8.0$$

```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```

s_2

$$R(s_2) = 1.2$$

- Maximize the reward over many generated summaries

Example from [CS288 slides]

Formalize a bit

- Treat LM as some distribution $p_\theta(s)$ over all possible summaries
- Maximize average reward over many generated summaries

$$\max_\theta \mathbb{E}_{s\sim p_\theta(s)}[R(s)]$$

- Different from the objective we saw before, why?
- Cannot be solved by SGD, but by policy gradient

Policy Gradient Descent

$$\frac{\partial \mathbb{E}_{s\sim p_\theta(s)}[R(s)]}{\partial\theta} = \frac{\partial}{\partial\theta}\int R(s)p_\theta(s)ds$$

$$= \int R(s)\frac{\partial p_\theta(s)}{\partial\theta}ds$$

$$= \int R(s)\frac{1}{p_\theta(s)}\cdot\frac{\partial p_\theta(s)}{\partial\theta}\cdot p_\theta(s)ds$$

$$= \mathbb{E}_{s\sim p_\theta(s)}[R(s)\cdot\nabla\ln p_\theta(s)]$$

$$\approx \frac{1}{m}\sum_{i=1}^{m}R(s_i)\cdot\nabla\ln p_\theta(s_i)$$

- Sample summaries s_i's from current $p_\theta(s)$

Note: The actual algorithm is used is PPO. We will revisit later!

On the Reward Function $R(s)$

- If we simply ask for numeric scores
  - hard to calibrate
  - costly
  - Human annotators suffer 🙁
- Instead we only ask for comparison

An earthquake hit San Francisco. There was minor property damage, but no injuries.	$>$	A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.	$>$	The Bay Area has good weather but is prone to earthquakes and wildfires.
s_1		s_3		s_2

Example from [CS288 slides]

Learn a function $R(s)$

- Bradley-Terry Model

$$p(s_i > s_j) = \frac{e^{R(s_i)}}{e^{R(s_i)} + e^{R(s_j)}} = \frac{1}{1 + e^{R(s_j) - R(s_i)}} = \sigma(R(s_i) - R(s_j))$$

- Parameterize the $R(s; \boldsymbol{w})$ as some network
- A binary classifier on event $s_i \lesseqgtr s_j$
- Denote $y_{i,j} = 1$ if $s_i > s_j$ else -1

$$\max_{\boldsymbol{w}} \sum_{i,j} \log \sigma\left(y_{i,j} \cdot \left(R(s_i; \boldsymbol{w}) - R(s_j; \boldsymbol{w})\right)\right)$$

Put together: Instruction finetuning + RLHF

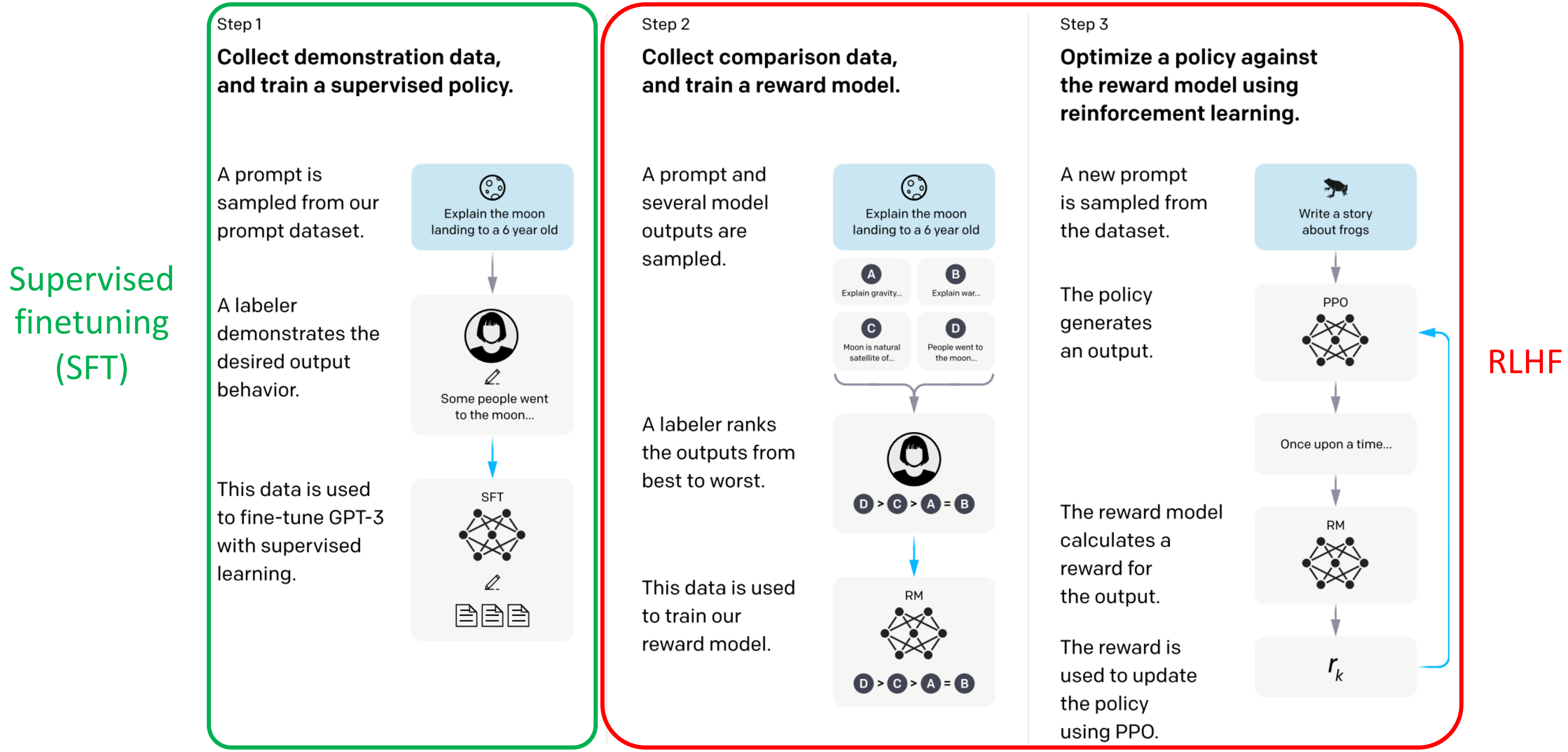
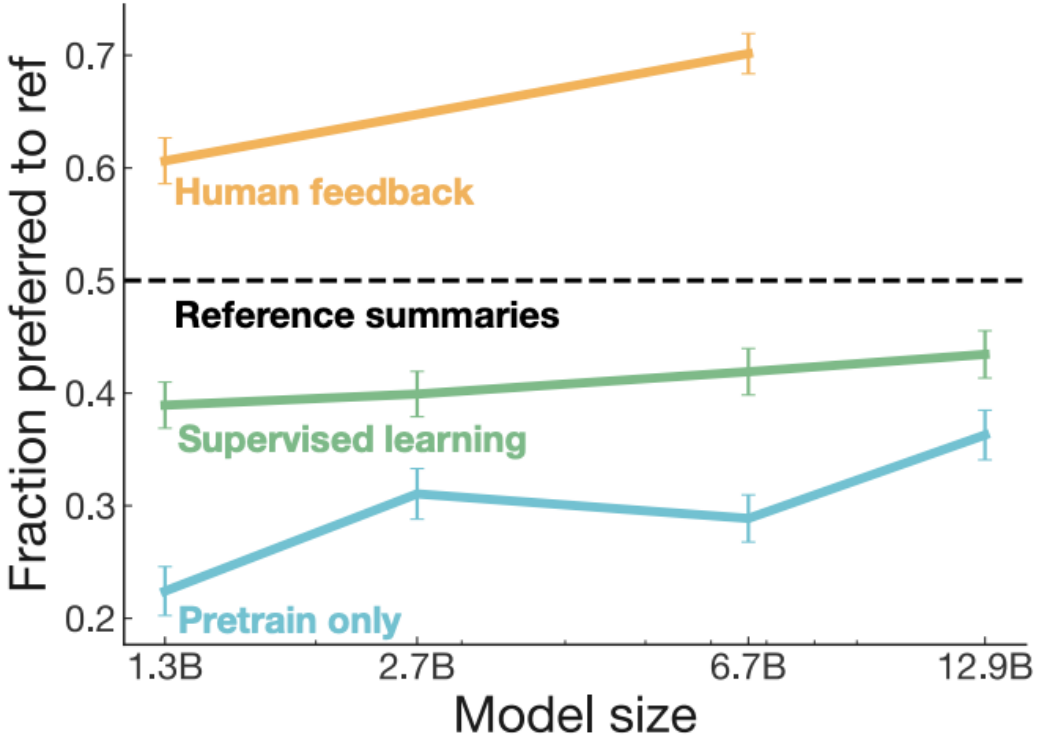


Figure from Ouyang et. al, 2022

Further Gain by RLHF



[Stiennon et. al, 2020]()