From **GPT** to **ChatGPT:** In-context Learning, Instruction Tuning, and Preference Optimisation

Antoine Bosselut

EPFL

Announcements

- **Midterm:** April 9th, next week!
- **Project:** Description document released after Midterm

Midterm Logistics

- **Date:**
  - Wednesday, April 9th, 2025
  - 11:30 - 13:00
- **Location:** 6 rooms on campus
  - INF 1, AAC 2 31, CE 1 3, CM 1 1, CM 1 2, CM 1 4
  - Each student will be pre-assigned to one of the rooms.
    - **YOUR EXAM COPY WILL BE IN THE ROOM WHERE YOU WERE ASSIGNED.**
    - **YOU MUST GO TO THE CORRECT ROOM.**
    - **NO EXTRA EXAMS WILL BE AVAILABLE IN OTHER ROOMS.**

- **Schedule**
  - **A few days before**: room assignments and seating chart released on Moodle
  - **11:00** on the day of: Exam rooms set up
  - **11:20** on the day of: Exam hall doors open and students take their seat.
  - **11:30** on the day of: Exam begins
  - **During exam**: EPFL ID Cards checked (**don’t forget CAMIPRO!**)
  - **13:00** on the day of: Exams collected

Midterm Format

- 30 Multiple Choice QA
  - From **lectures** AND **exercise sessions (up to and including Week 7)**
  - **No** negative points on MCQA
- Materials
  - 1 double-sided A4 crib sheet allowed
  - Non-programmable calculators allowed
  - CAMIPRO card
- No phones or other electronic devices allowed

Midterm Office Hour

- Class period (13h15 - 14h) on **April 3rd (tomorrow)**
- Come with questions!

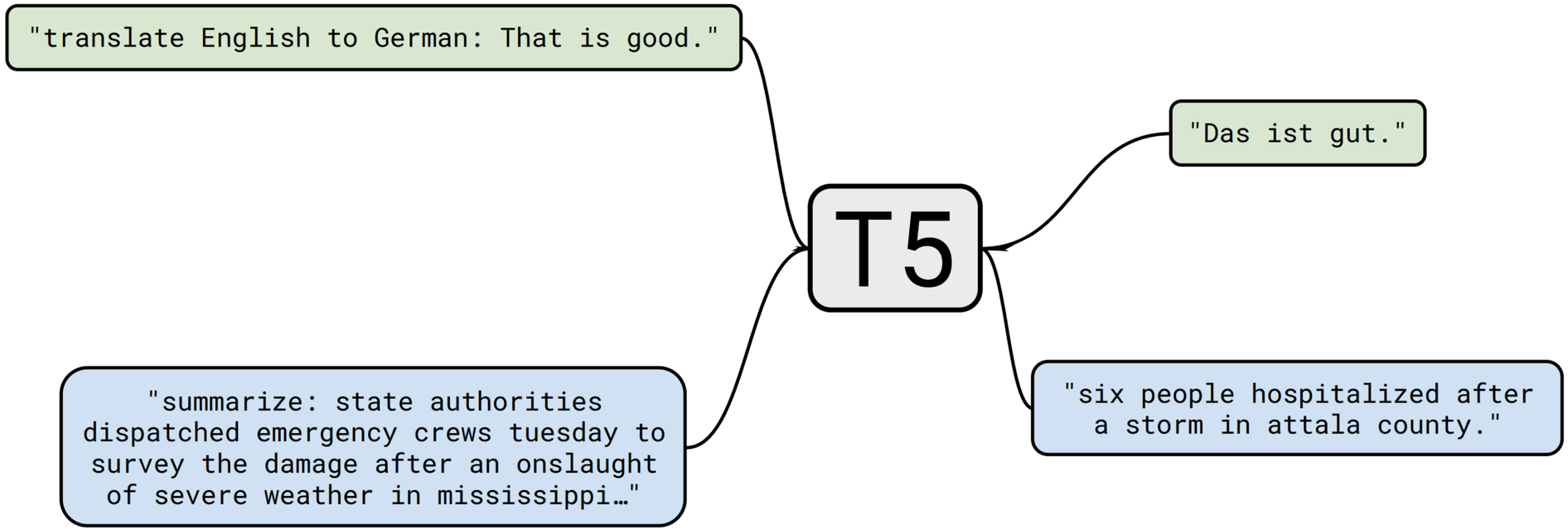
Today's Outline

- **Lecture**
  - **Quick Recap:** Fine-tuning, T5
  - **GPT-3:** Scaling, Emergent Behavior, In-context Learning, Chain-of-Thought, ChatGPT
- **Tomorrow:**
  - **Midterm Review Session (**1:15 PM @ CE 1 6**)**
  - **Week 7 Exercise Session:** In-context learning (2:15 PM @ CE 1 1)
  - **Review of Week 6 Exercise Session:** Text Generation

Some slides adapted from Mohit Iyyer

T5

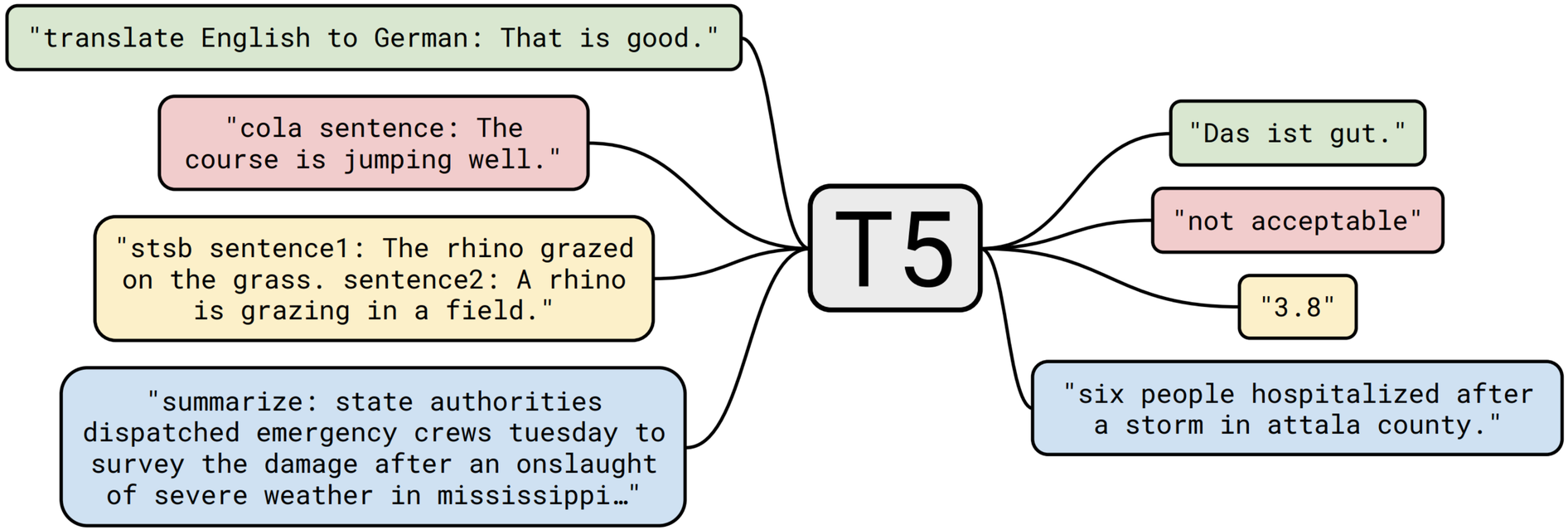
Any problem can be cast as sequence-to-sequence



Raffel et al. (2019)

T5

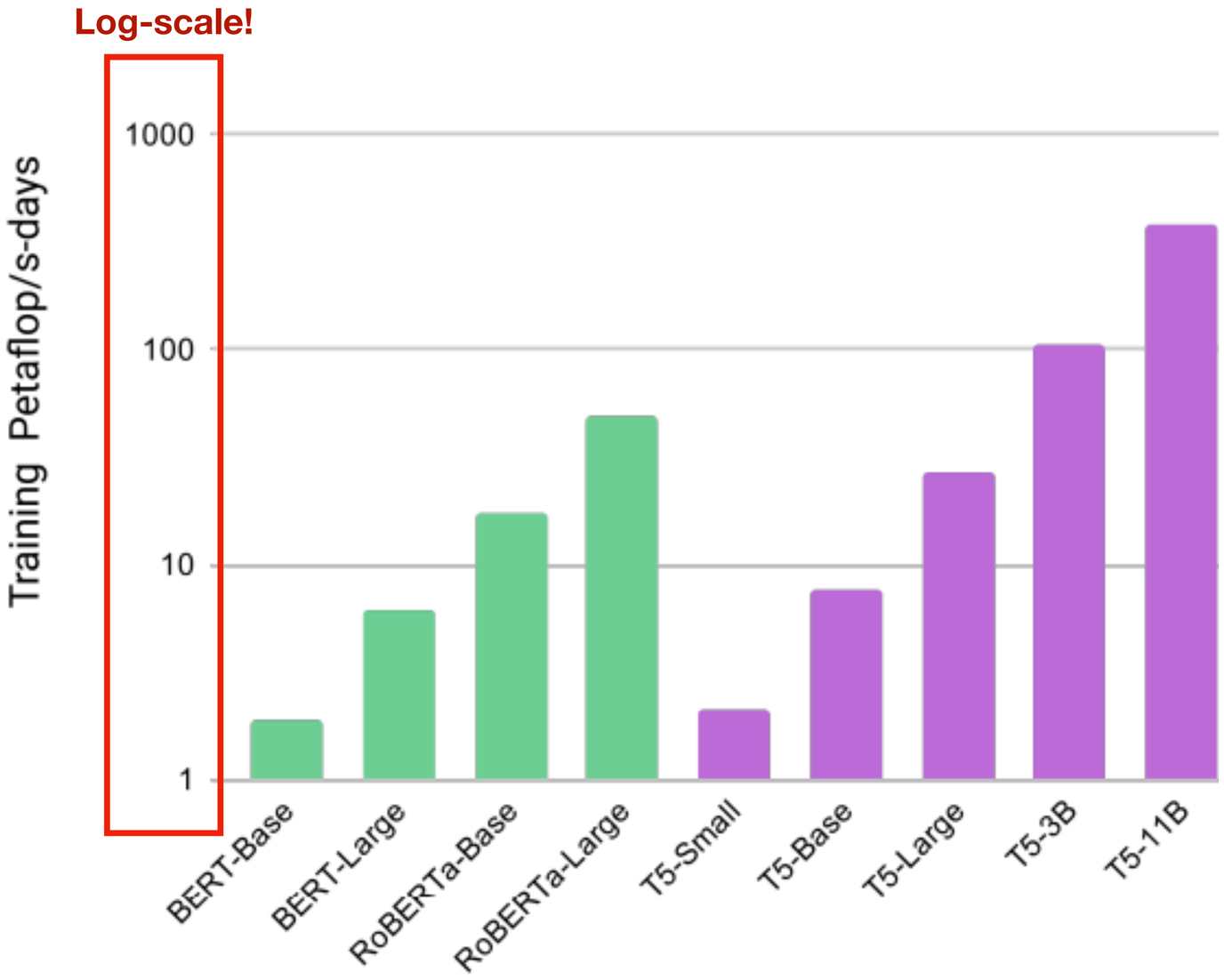
Any problem can be cast as sequence-to-sequence



Rather than learning from the output embeddings of a [CLS] token, learn to generate the language that represents the semantics of the label

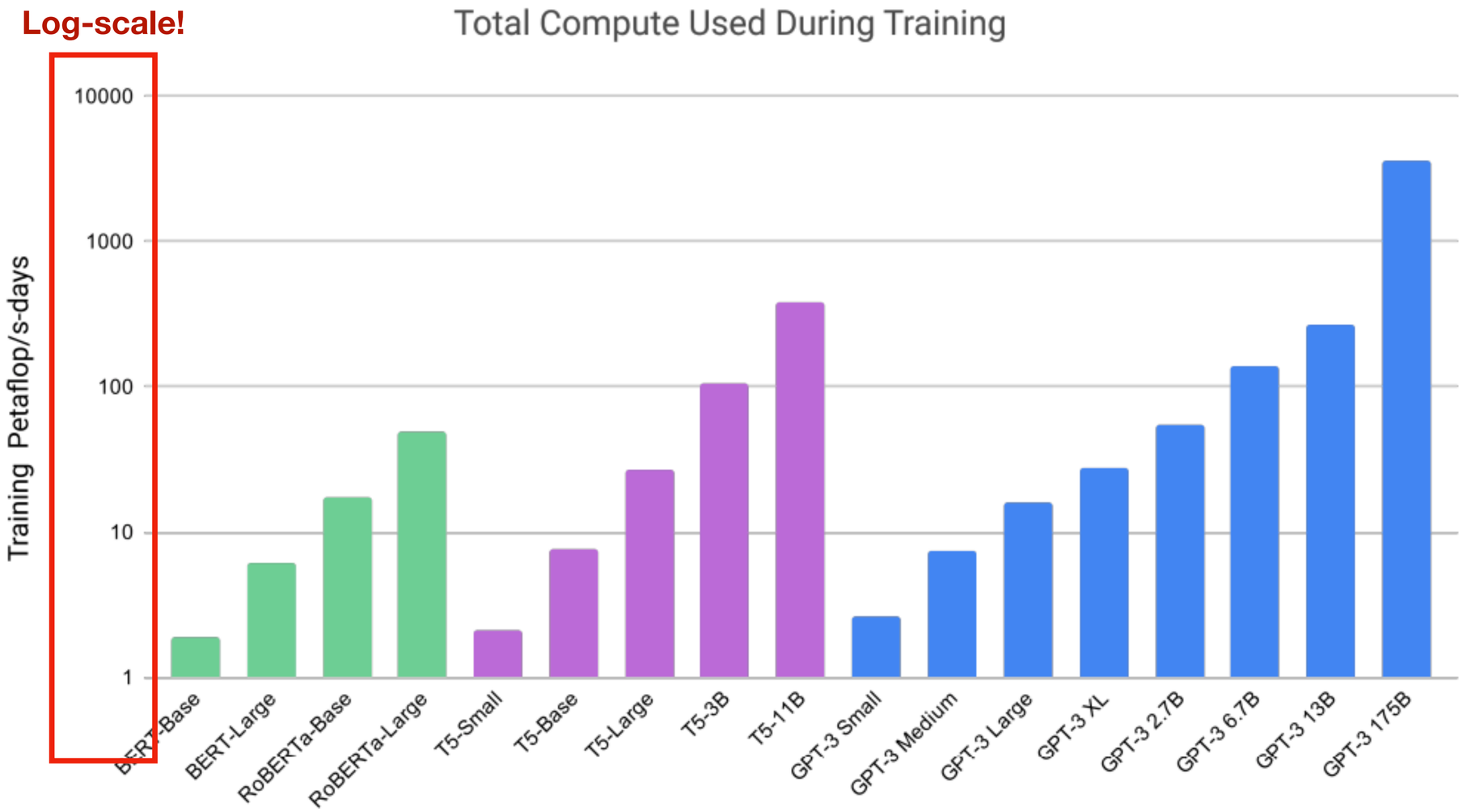
Raffel et al. (2019)

Language Model Scaling

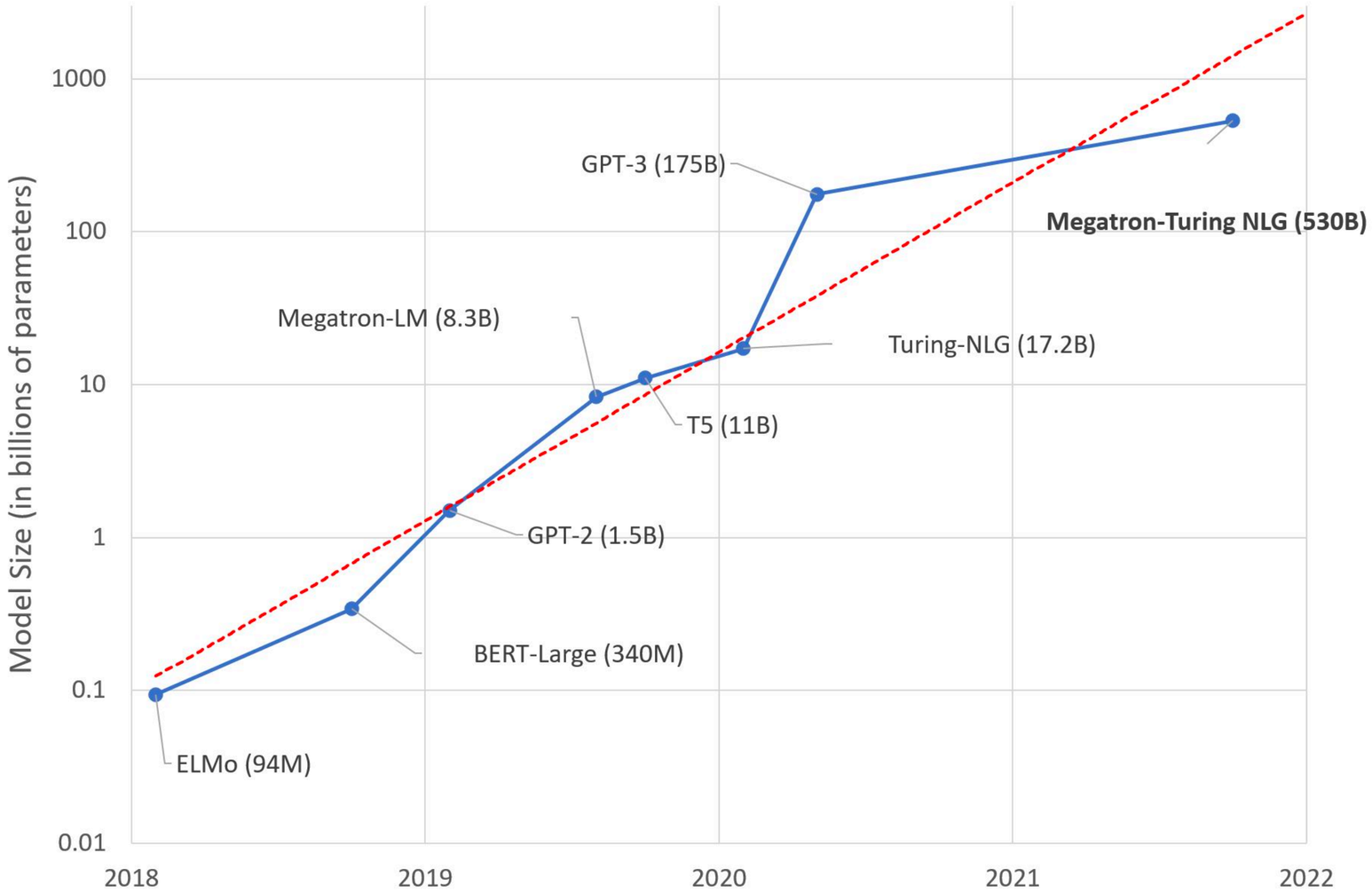


Much larger scale than before!

Language Model Scaling



Language Model Scaling



Larger models

More data

More compute

More

What happens when we train at this very large scale ?

“Emergence”

“Emergence is when quantitative changes in a system result in qualitative changes in behavior.”

- Philip Anderson, 1972

"Emergence"

"Emergence is when quantitative changes in a system result in qualitative changes in behavior."

in physics: atoms → molecules → life

- Philip Anderson, 1972

For LLMs, "an ability is emergent if it is not present in smaller models but is present in larger models."

LLMs: tiny model → can't do logic
big model → can

Controversial Topic!

Wei et al. (2022)

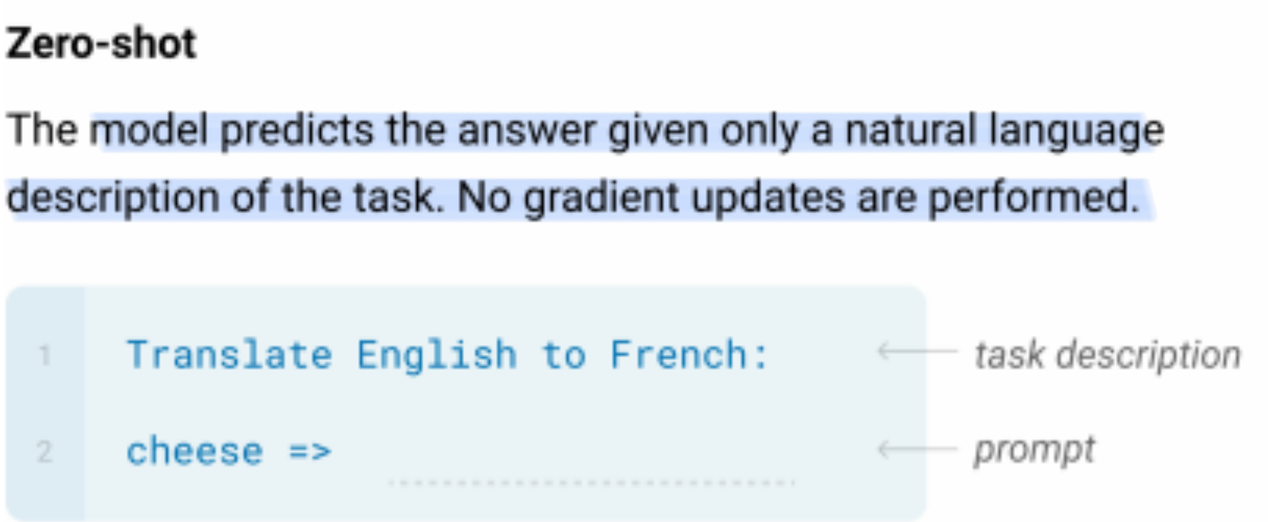
In-context Learning

Old Paradigm: Fine-tuning (GPT, BERT)

- In fine-tuning, we:
  - provide training examples to our model and predict labels
  - compute gradients of the loss of these predictions w.r.t. all parameters in the model
  - update the parameters of the model based on these gradients
  - Once the model is trained, we test on other examples we didn't train on
- The model learns by **updating its parameters** to better predict the examples it sees during training

Brown et al. (2020)

Old Paradigm: Zero-shot (GPT2)



- In zero-shot adaptation, we use the pretrained model to complete tasks
  - The command we give the model to do a task: **task description**
  - The way we frame the example: **prompt**

Brown et al. (2020)

New Paradigm: In-context Learning

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

- In in-context learning, we concatenate the following in the **same** input sequence:
  - A task description (as in zero-shot learning)
  - An example (or multiple examples) of the task (i.e., context + label)
  - The final prompt, for which we want the model to predict a label

Brown et al. (2020)

New Paradigm: In-context Learning

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

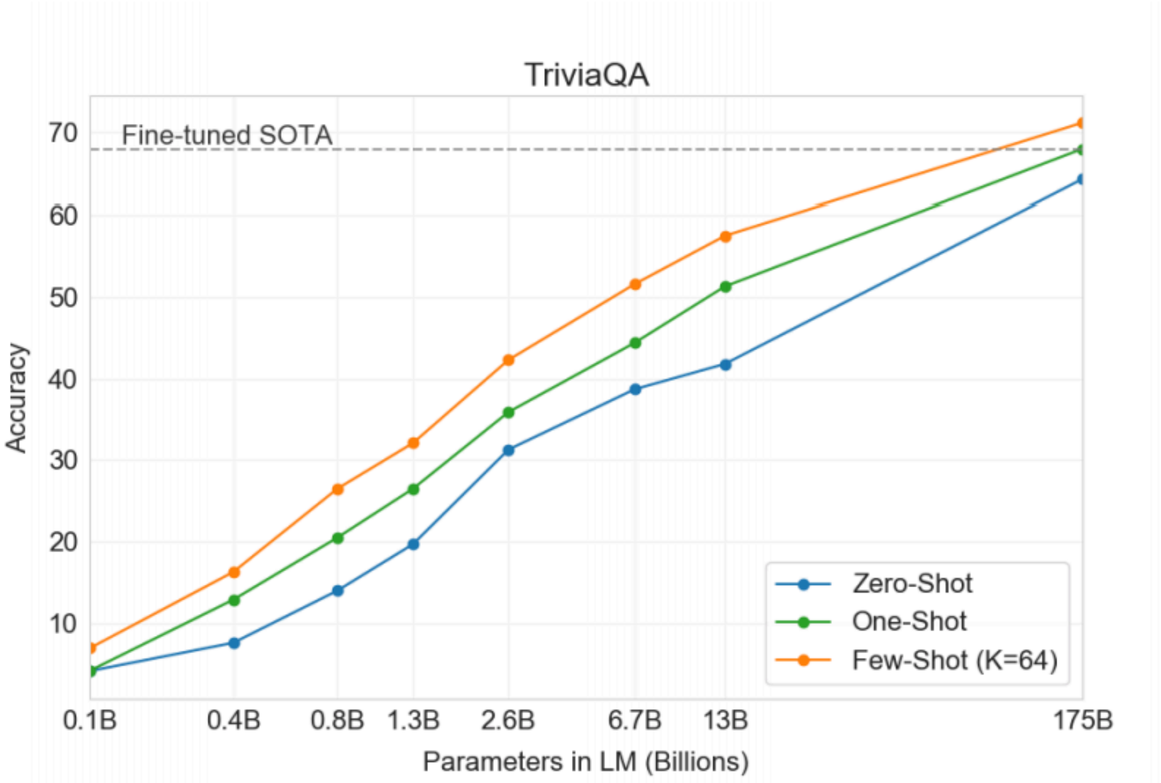
There is no more learning in the classical sense!

- In in-context learning, we concatenate the following in the **same** input sequence:
  - A task description (as in zero-shot learning)
  - An example (or multiple examples) of the task (i.e., context + label)
  - The final prompt, for which we want the model to predict a label
- The model "learns" by attending to the tokens in the in-context examples

Brown et al. (2020)

TriviaQA Results

- Questions like:
  - Miami Beach in Florida borders which ocean?
  - What was the occupation of Lovely Rita according to the song written by the Beatles?
  - What was the Elephant man's real name?
  - Which type of hat takes its name from an 1894 novel by George Du Maurier where the title character has the surname O'Farrell?
  - Who was Poopdeck Pappy's most famous son?

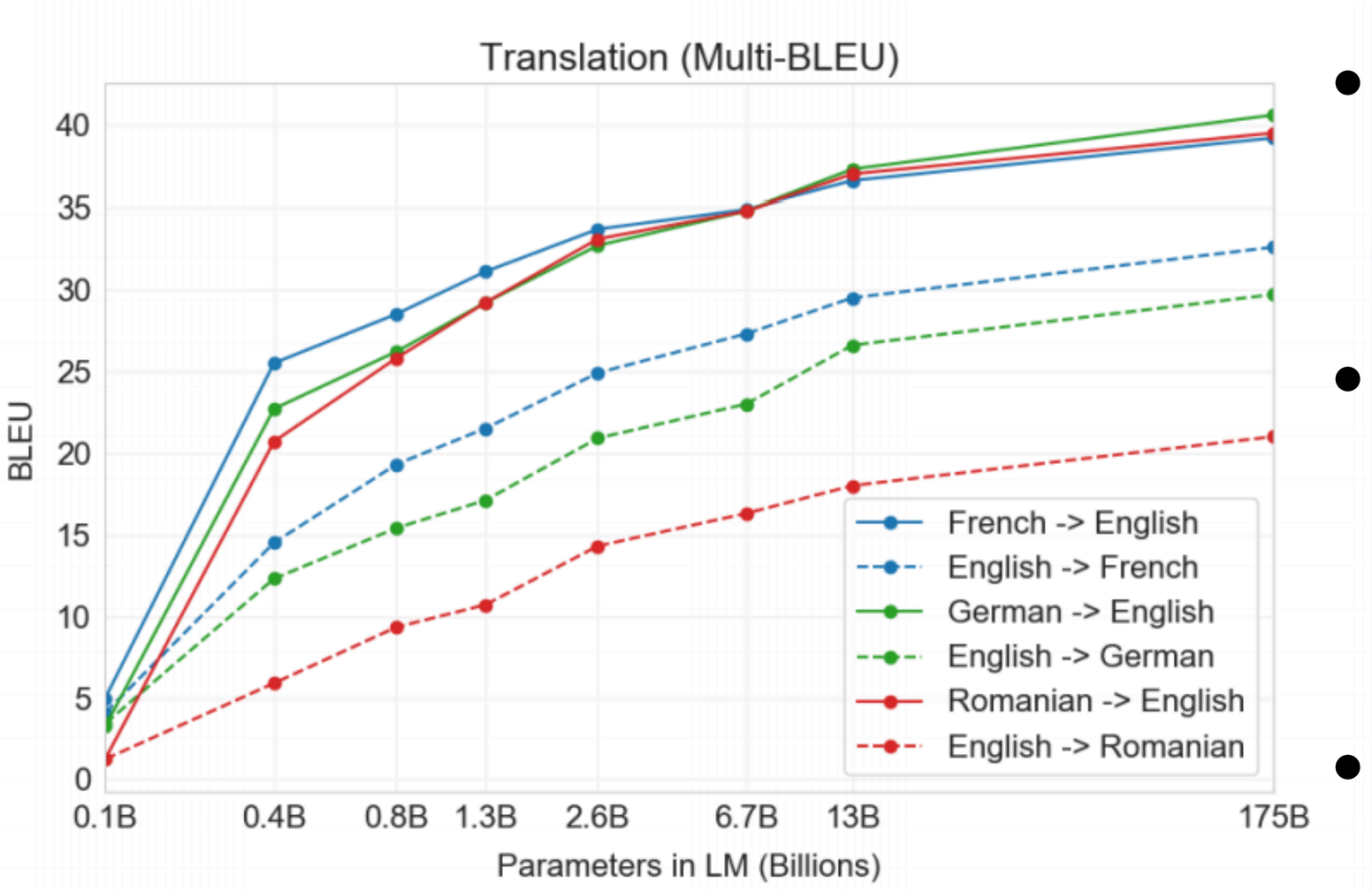


No access to supporting documents!

Only memorisation!!!

Brown et al. (2020)

Machine Translation



- Model not explicitly trained to translate from one language to another
- Model not **purposefully** trained on multilingual data
  - 7% of data is not English, though
- Model manages to learn **some** translation abilities

Brown et al. (2020)

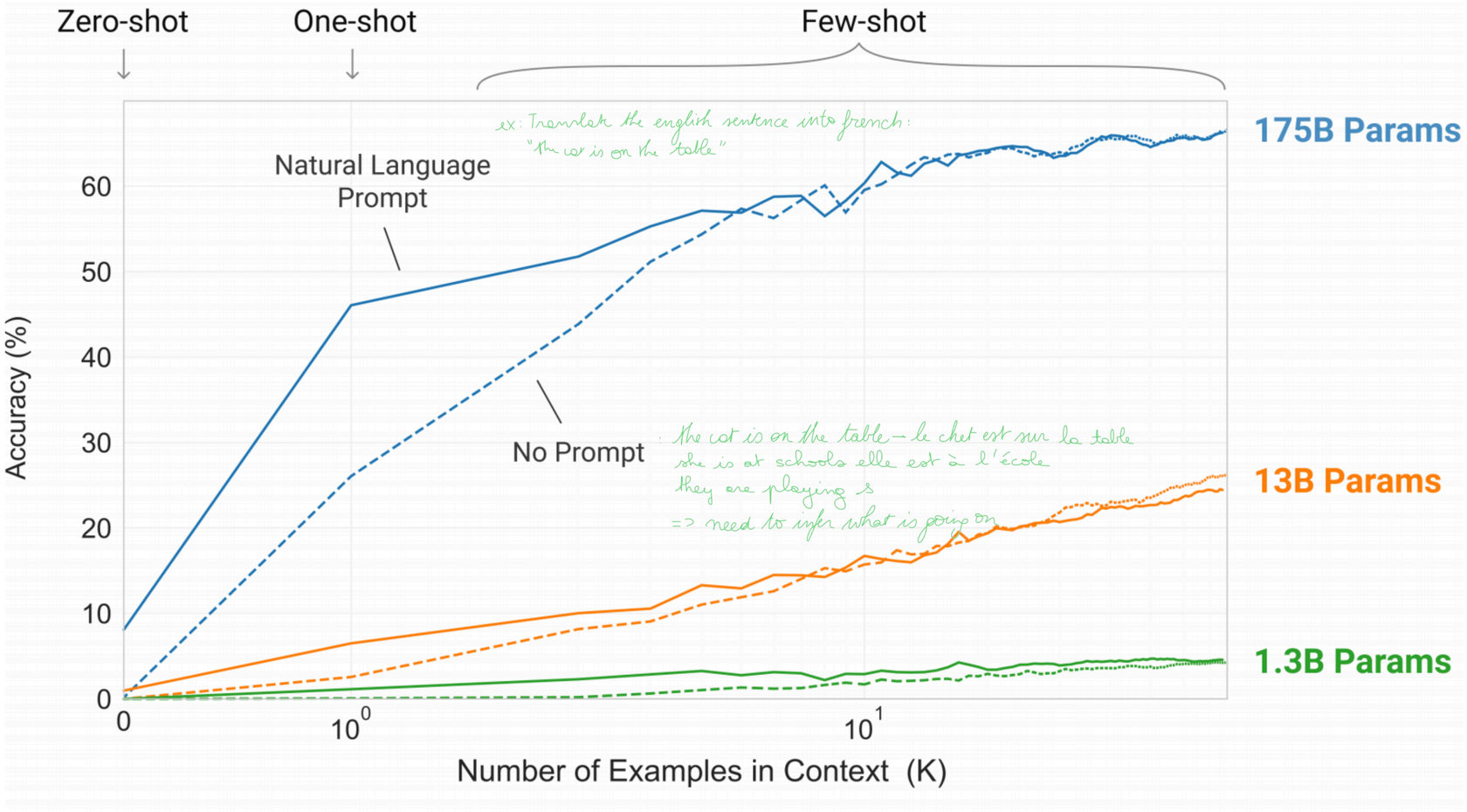
Machine Translation

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Model much better at translating into English

Brown et al. (2020)

Emergence: Necessity of Scale



~ 10X T5 scale

~ T5 scale (largest model we normally fine-tune)

~ GPT2 scale

In-context learning only works well with VERY large models!

Brown et al. (2020)

In-context Learning is sensitive

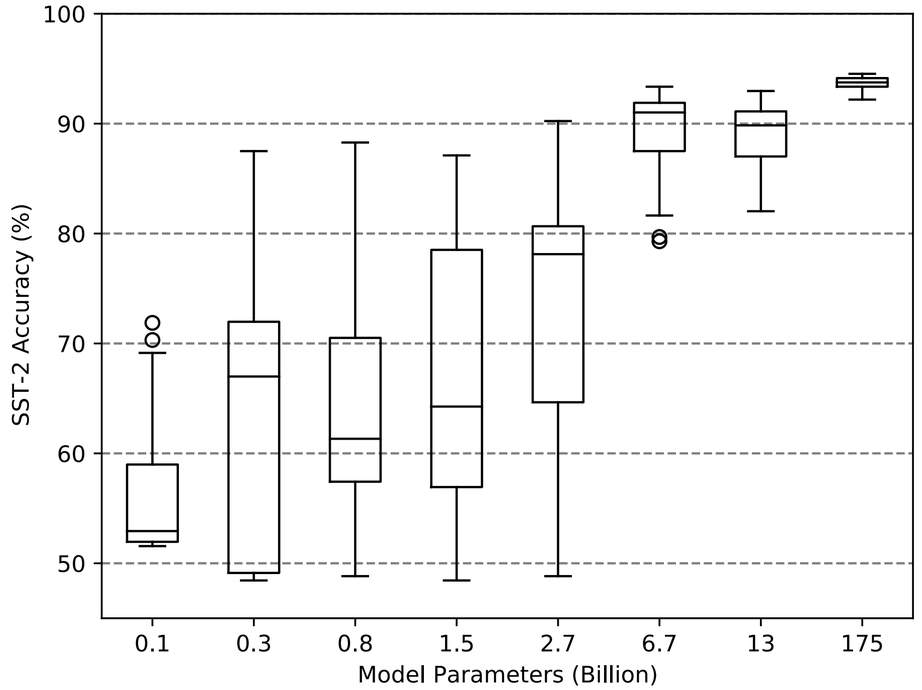
Trial	1	2	3	4	5
Accuracy	94.6	95.0	95.8	93.9	86.9

Table 1: Results of GPT-3 on the task of sentiment analysis on the SST-2 dataset. Five different in-context examples are randomly selected from the training set. We observe different contexts induce different accuracies on the test set.

Liu et al., (2021)

Example selection matters!

Performance distribution across 24 random orders of in-context examples



Lu et al., (2021)

Large variance in performance.

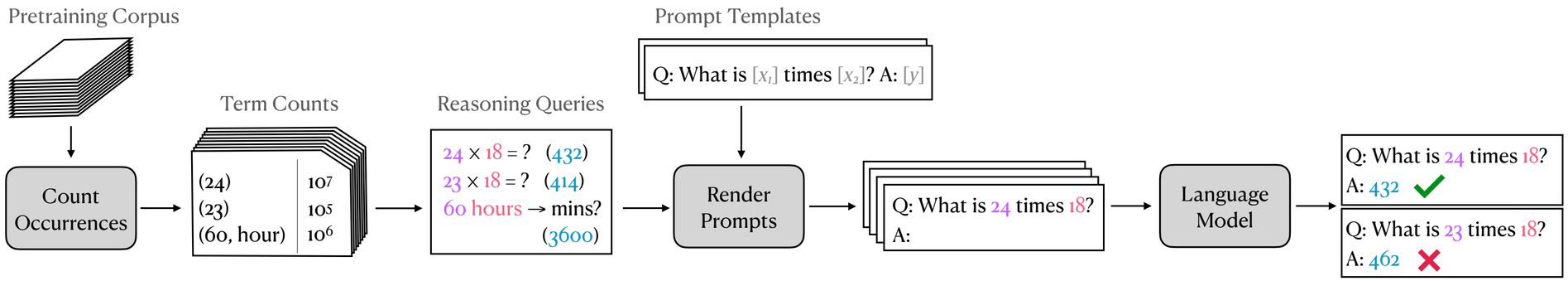
Example order matters!

Is less present in larger models

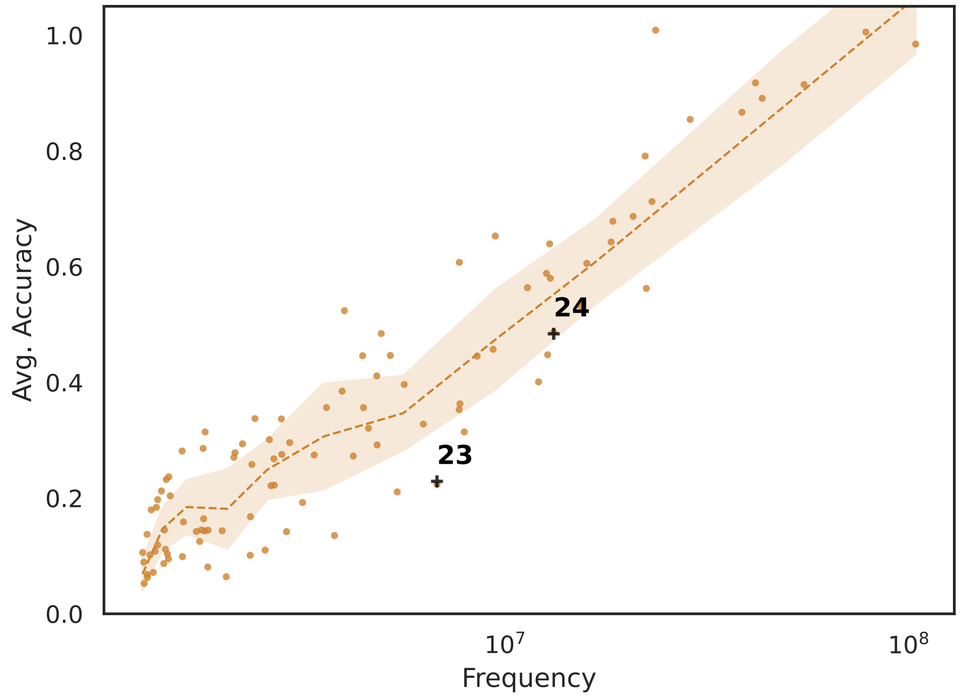
=> familiarity helps : more a model has seen a term the better it performs with it in in-context tasks
=> no full theory yet

Why does in-context learning work ?

=> data matters : what's in pretraining corpus strongly influences model's ability to learn new tasks



- No clear & unifying theory of why in-context learning works, yet
- Generally, terms that are **more frequently mentioned** in pretraining corpora are easier for models to parse for in-context learning



How can we make in-context learning more effective?

Prompt Engineering

Yelp For the Yelp Reviews Full Star dataset (Zhang et al., 2015), the task is to estimate the rating that a customer gave to a restaurant on a 1- to 5-star scale based on their review's text. We define the following patterns for an input text a:

$P_1(a) =$ It was ____. a $\quad P_2(a) =$ Just ____! $\|$ a

$P_3(a) =$ a. All in all, it was ____.

$P_4(a) =$ a $\|$ In summary, the restaurant is ____.

"patterns" for expressing labels

We define a single verbalizer v for all patterns as

$v(1) =$ terrible $\quad v(2) =$ bad $\quad v(3) =$ okay

$v(4) =$ good $\quad v(5) =$ great

"verbalizer" of actual labels

✅ **Step 2: Feed this to the model**

This is your full input (the prompt):

```
arduino
"It was ____ a. The food was decent, but the service was slow and the place was too
```

The model is expected to fill in the blank with one of the **verbalizers**:

- terrible (1 star)
- bad (2 stars)
- okay (3 stars)
- good (4 stars)
- great (5 stars)

✅ **Step 3: Interpret the model's output**

Suppose the model outputs:

```
arduino
"It was bad a. The food was decent, but the service was slow and the place was too
```

You then map **"bad" → 2 stars** based on the verbalizer:

```
v(2) = bad
```

Schick and Schutze, (2020)

= process of crafting inputs in a smart way to get better performance from a LM without changing the model itself

Prompt Engineering

LMs are better at generating words than labels or numbers → reframe task using natural sounding patterns
↘ use words (verbalizers) the model understands

Yelp For the Yelp Reviews Full Star dataset (Zhang et al., 2015), the task is to estimate the rating that a customer gave to a restaurant on a 1- to 5-star scale based on their review's text. We define the following patterns for an input text a:

$P_1(a) =$ It was ____. a $\quad P_2(a) =$ Just ____! $\|$ a

$P_3(a) =$ a. All in all, it was ____.

$P_4(a) =$ a $\|$ In summary, the restaurant is ____.

"patterns" for expressing labels

We define a single verbalizer v for all patterns as

$v(1) =$ terrible $\quad v(2) =$ bad $\quad v(3) =$ okay

$v(4) =$ good $\quad v(5) =$ great

"verbalizer" of actual labels

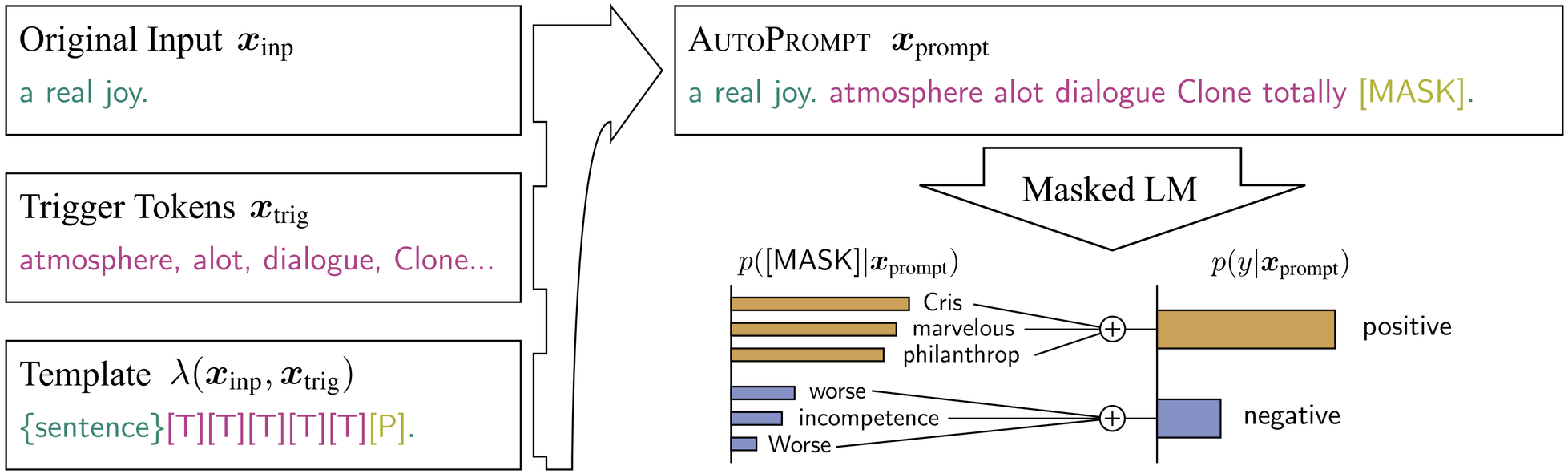
- Select the patterns and verbalisers that make in-context learning more effective at completing the task

Prompts can even sometimes improve performance at small-scale too!

Schick and Schutze, (2020)

What makes for the best prompt?

- **AutoPrompt**: Search for the pattern tokens that make the best prompt
  - **Gradient-guided search**: initialise a set of trigger tokens for the prompt and estimate the change in loss from replacing trigger tokens with other tokens in the vocabulary



Shin et al., (2020)

What makes for the best prompt?

Task	Prompt Template	Prompt found by AUTOPROMPT	Label Tokens
Sentiment Analysis	{sentence} [T]. . . [T] [P].	unflinchingly bleak and desperate Writing academicswhere overseas will appear [MASK].	**pos**: partnership, extraordinary, ##bla **neg**: worse, persisted, unconstitutional
NLI	{prem}[P][T]. . . [T]{hyp}	Two dogs are wrestling and hugging [MASK] concretepathic workplace There is no dog wrestling and hugging	**con**: Nobody, nobody, nor **ent**: ##found, ##ways, Agency **neu**: ##ponents, ##lary, ##uated
Fact Retrieval	*X plays Y music* {sub}[T]. . . [T][P].	Hall Overton fireplacemade antique son alto [MASK].	
Relation Extraction	*X is a Y by profession* {sent}{sub}[T]. . . [T][P].	Leonard Wood (born February 4, 1942) is a former Canadian politician. Leonard Wood gymnasium brotherdicative himself another [MASK].	

Sometimes, sequences of random words!

Foundation of "Jailbreaking"

Shin et al., (2020)

Optimal prompt search yields random sequences of tokens (i.e., vectors)

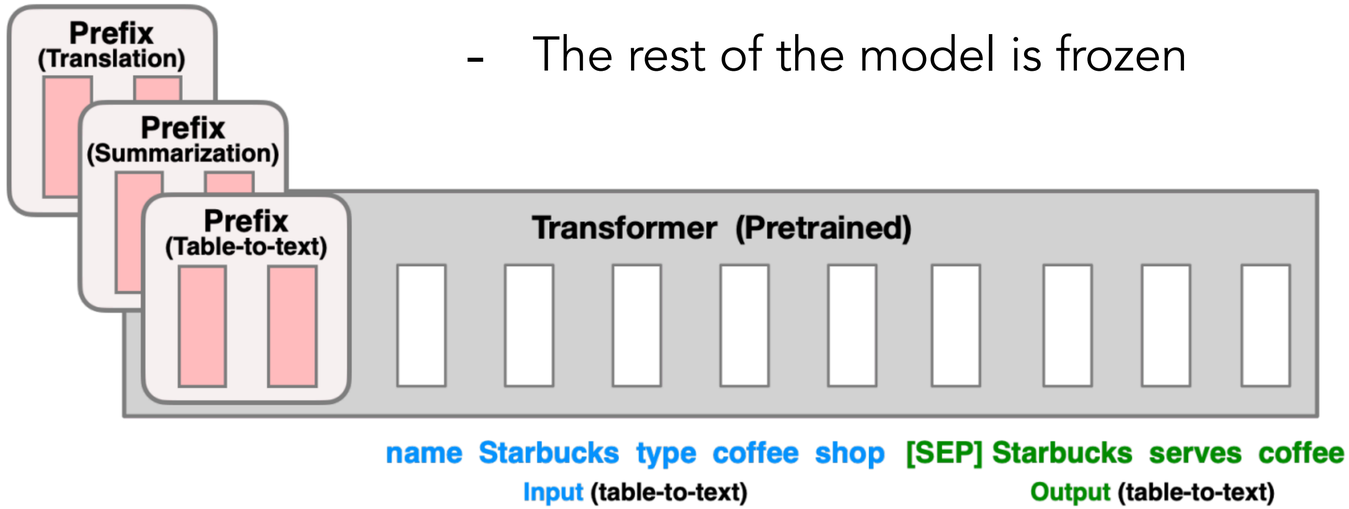
What else could we do?

Learn the prompt!

"Soft prompts" or Prompt Tuning

=> not words, made of learned embeddings (= vectors), as humans we aren't able to understand them + contrary to before they are learnt by the model
↳ they guide a language model

- For each task we want to do:
  - Initialise a special input prompt vector (or multiple prompt vectors)
  - Prepend them to any example of the task
  - Compute gradients of the loss with respect to the parameters of the prompt
  - Only the prompt vectors are updated by these gradients
  - The rest of the model is frozen

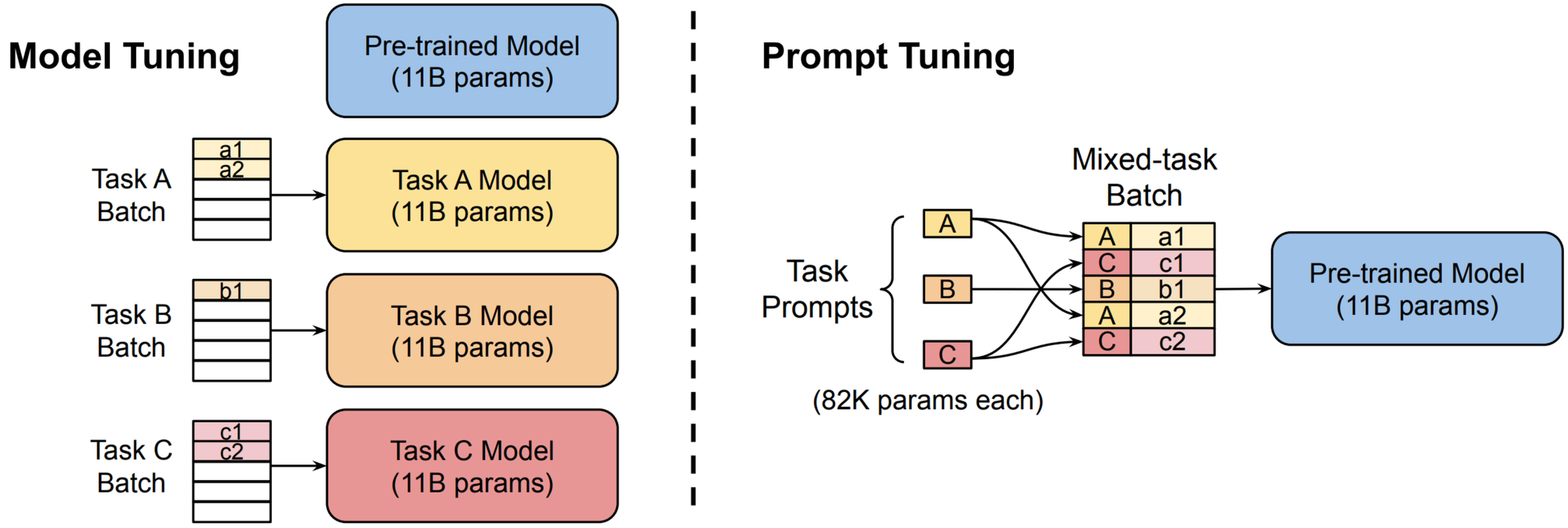


PIPELINE:
1) start with frozen model (GPT, BERT)
2) add small # of trainable vectors (= soft prompt embeddings) at the input level
3) input: [soft prompt] + [tokenized user inputs]
4) during training, only soft prompts get updated
→ model's weights stay frozen

Li and Liang, (2020)

What might be some
(dis)advantages of prompt tuning?

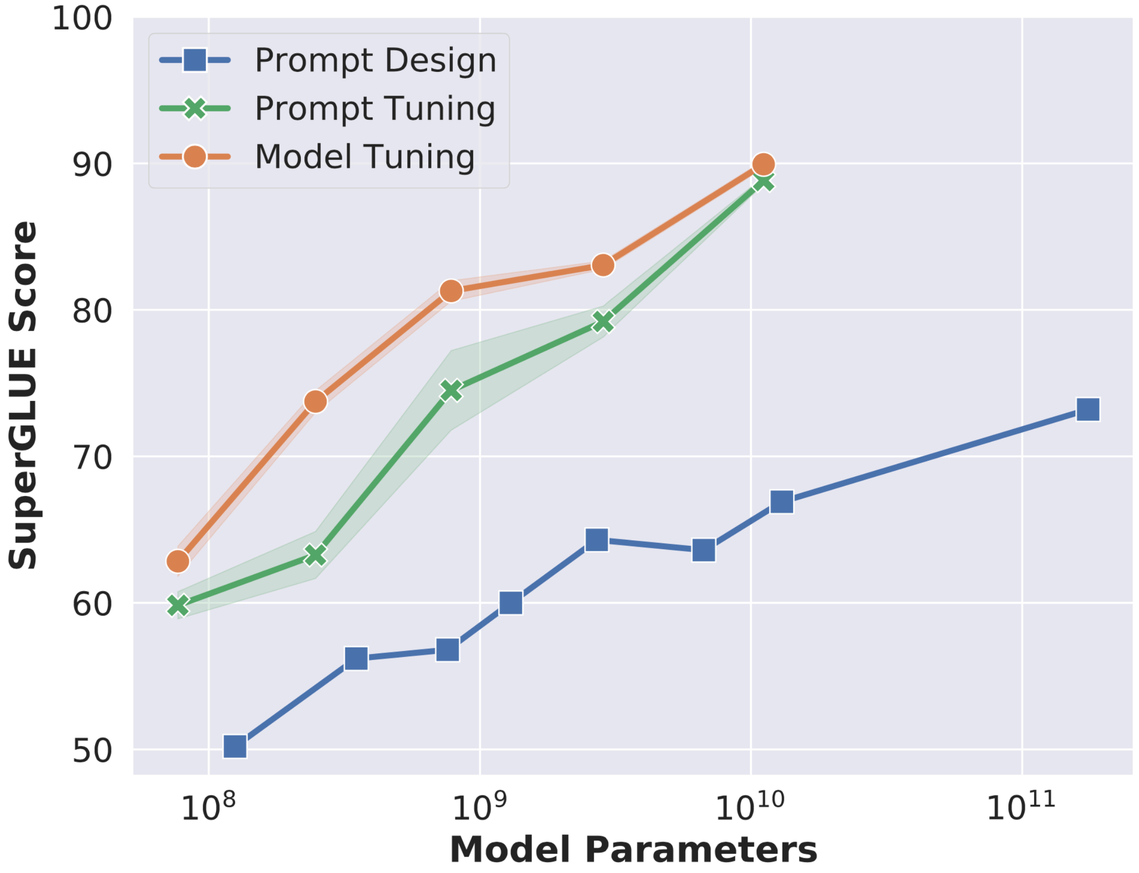
“Soft prompts” or Prompt Tuning



- Fine-tuning: update a new version of the model for each task
- Prompt tuning: update the prompt vectors for each task. **Other parameters are frozen.**
  - More efficient training (and separable multi-task learning)
  - Not as interpretable as discrete prompts!

Lester et al., (2021)

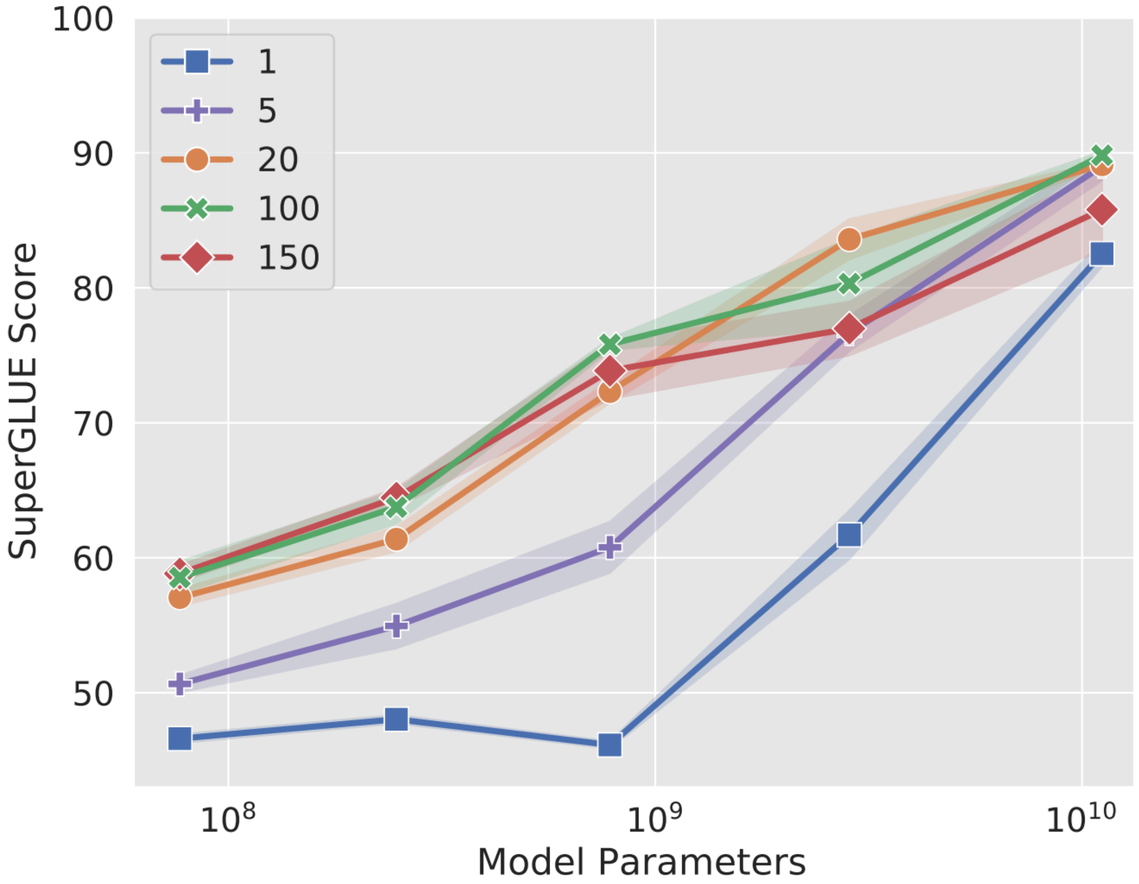
Prompt Tuning



- As scale increases:
  - Prompt tuning becomes competitive with full model fine-tuning!

Lester et al., (2021)

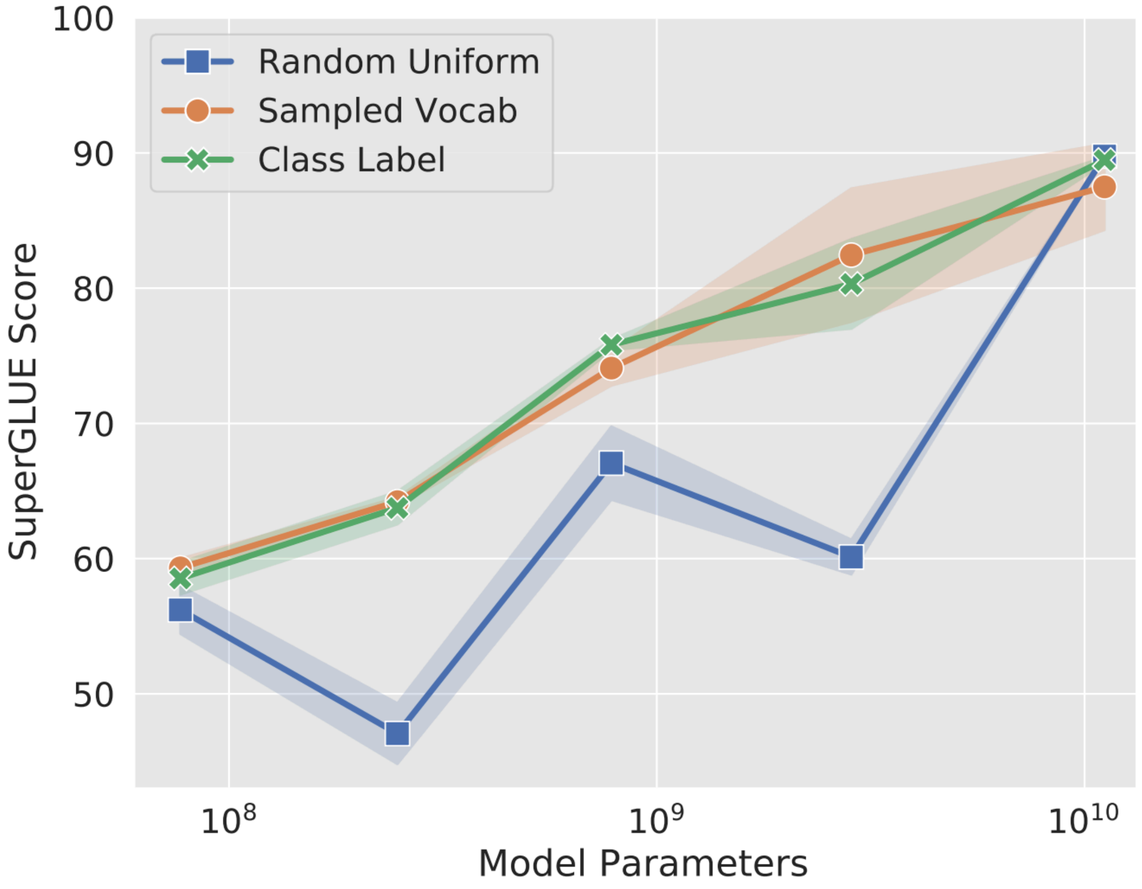
Prompt Tuning



- As scale increases:
  - Prompt tuning becomes competitive with full model fine-tuning!
  - The length of the prompt has less of an impact on the performance

Lester et al., (2021)

Prompt Tuning

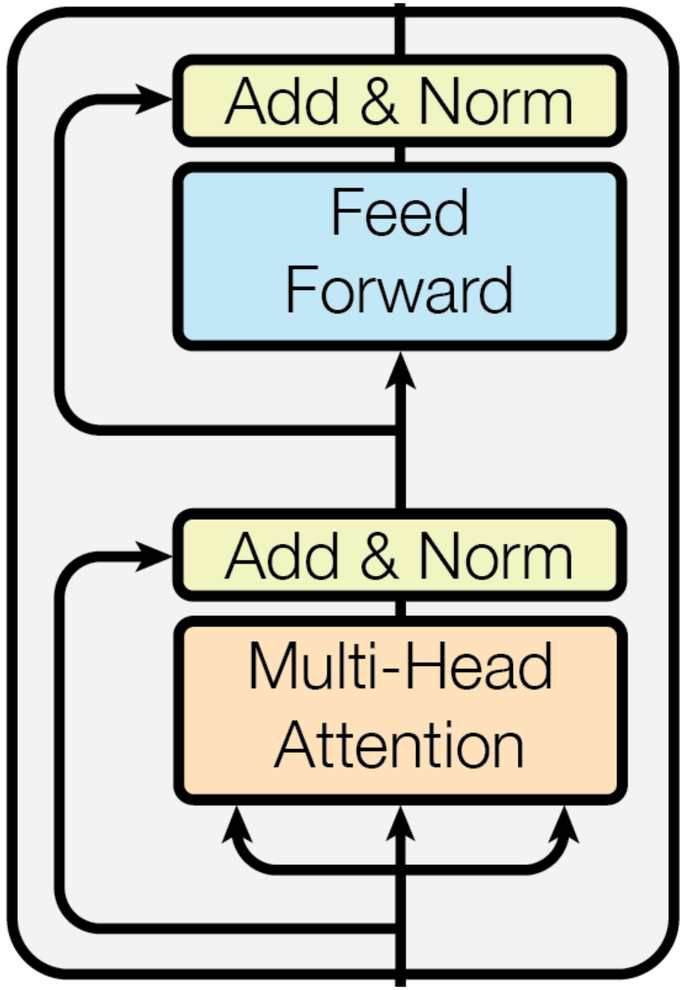


- As scale increases:
  - Prompt tuning becomes competitive with full model fine-tuning!
  - The length of the prompt has less of an impact on the performance
  - The initialisation of the prompt no longer matters

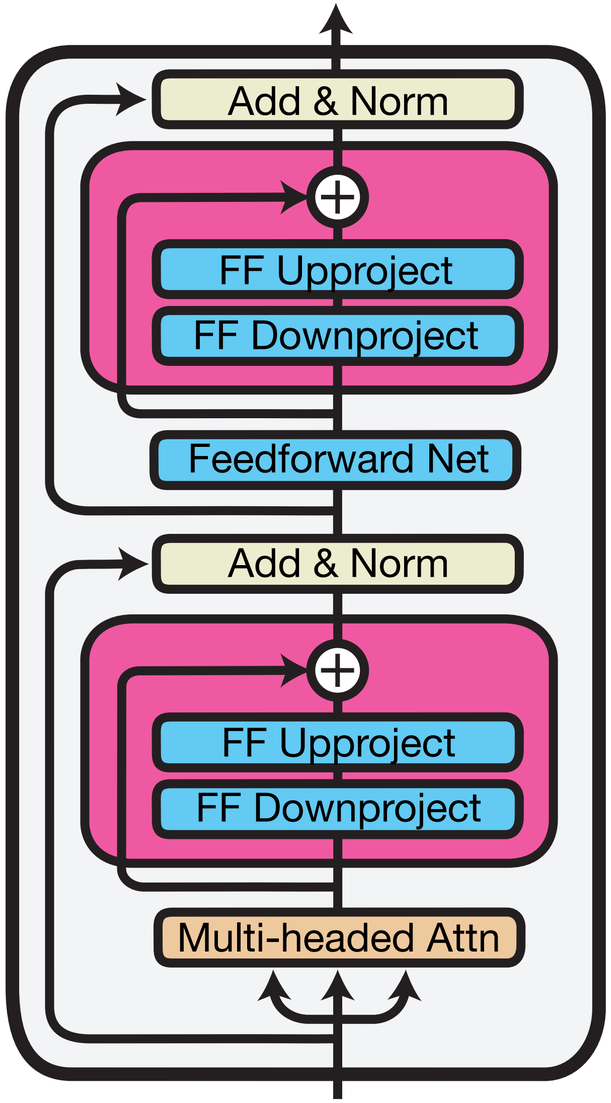
Need large models for these benefits to emerge!

Lester et al., (2021)

Efficient tuning: Adapters



Original Transformer Block

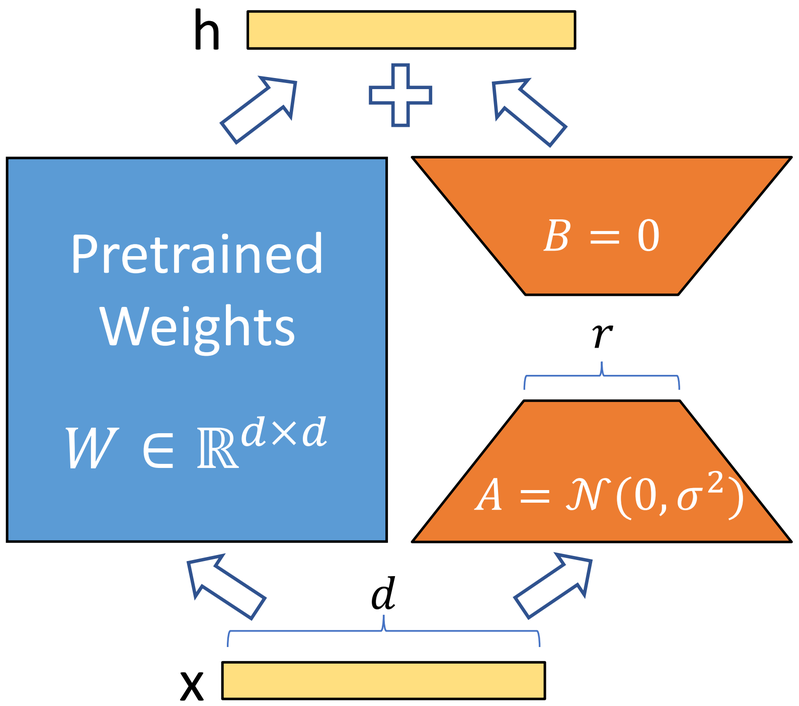


Houlsby Adapter

- During fine-tuning:
  - Keep all pretrained parameters frozen
  - **Adapters**: Initialise new **FFN layers** between components of transformer blocks
  - Keep these **FFN layers** limited in number of parameters
  - Only update these **FFN layers**

Houlsby et al., (2019)

Efficient tuning: LoRA



- During fine-tuning:
  - Keep all pretrained parameters frozen
  - **LoRA**: Initialise new **Feedforward Net (FFN)** alongside components of transformer blocks
  - Keep these **FFN layers** limited in number of parameters
    - # parameters in **FFN layers** is 2 * *d* * *r*, so keep *r small*
    - *r* is hidden dimension of **FFN**
  - Only update these **FFN layers**

Hu et al., (2021)

Quick Recap

- Abilities of language models emerge at scale: in-context learning
  - Can be improved using prompt engineering & prompt tuning
    - manual prompt design (Brown et al., 2020; Schick and Schutze, 2021)
    - Mining and paraphrasing to augment prompt sets (Jiang et al., 2020)
    - Gradient-based search for finding prompts (Shin et al., 2020)
    - Prompt generation using language models (Gao et al., 2020)
    - Soft prompts (Li and Liang, 2021; Lester et al., 2021, others)

- These approaches leverage the fact that:
  - Models are a lot bigger, and trained on more data for longer
  - All tasks can be mapped to a natural language description: predicting a label just involves producing the word we map as the label.

Guide the model to perform tasks using natural language descriptions

What are other model abilities become apparent at large scale?

Standard Prompting

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. 

Wel et al. (2022)

Chain-of-Thought Prompting

Standard Prompting

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

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. ✓

Wel et al. (2022)

Chain-of-Thought Prompting

- For complex problems:
  - Don’t just show the model the example prompt and the answer
  - Demonstrate the reasoning process behind the answer as part of the in-context examples
  - Model “learns” to produce an explanation as it decodes the answer

Chain of Thought Prompting

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.

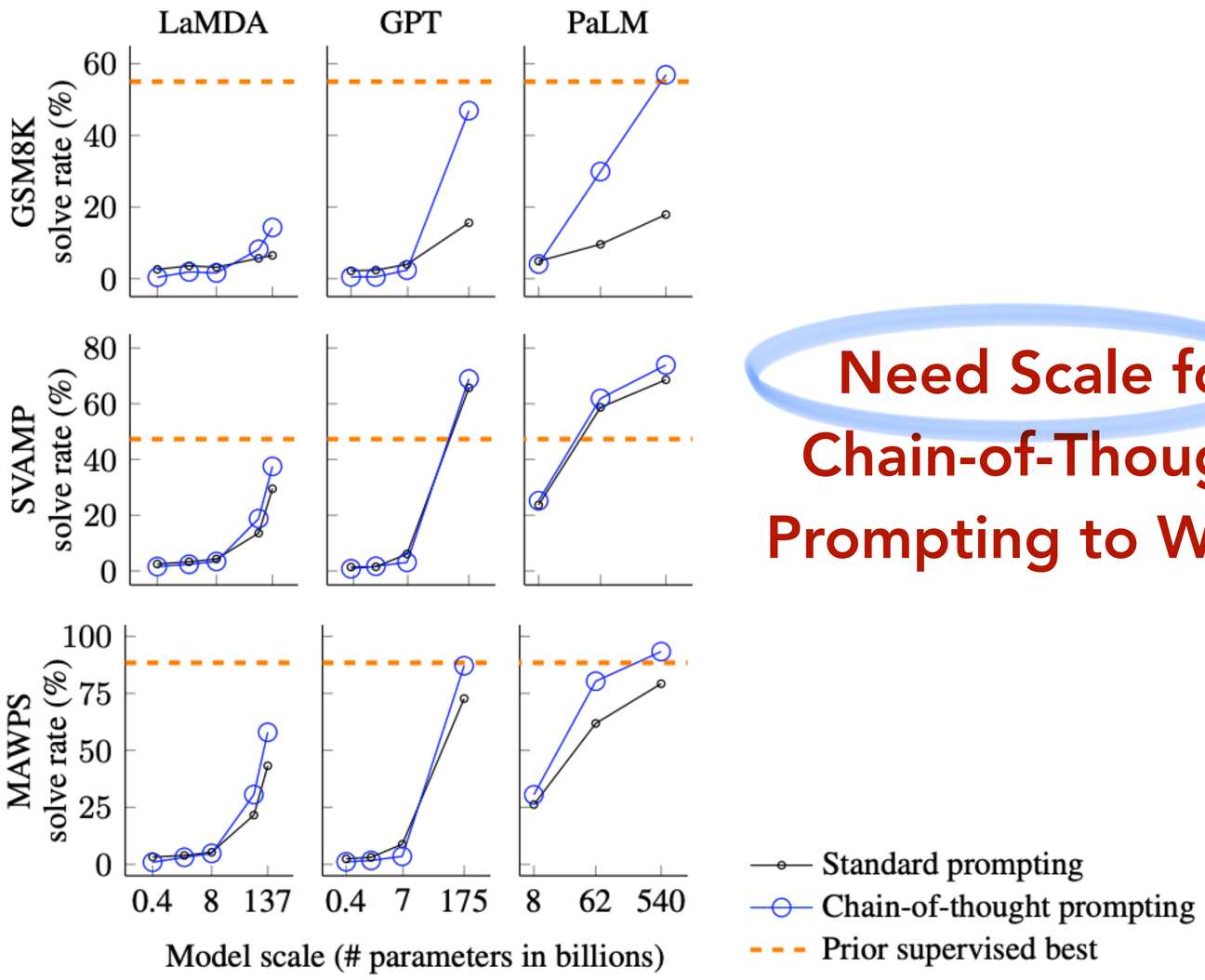
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

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. ✓

Wel et al. (2022)

Chain-of-Thought Prompting



Need Scale for Chain-of-Thought Prompting to Work!

Chain of Thought Prompting

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. ✓

Wel et al. (2022)

What do those two abilities remind you of?

ChatGPT!

AN

What language model is capable of following instructions and providing explanations for its answers ?

As an AI language model, I am an example of a system that is capable of following instructions and providing explanations for my answers. I am based on the GPT-4 architecture, developed by OpenAI. The GPT series of models, including GPT-3 and GPT-4, are designed to generate coherent and contextually relevant text based on given prompts, such as instructions or questions.

These models can understand and respond to a wide range of requests, including providing explanations for their responses. However, it is important to note that AI language models like GPT-4 are not perfect and can sometimes generate incorrect or incomplete information. As AI continues to evolve, newer and more advanced models will likely become even better at following instructions and providing accurate explanations.

Going from GPT-3 to ChatGPT

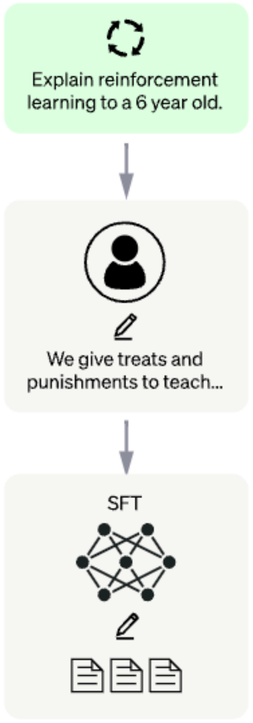
Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



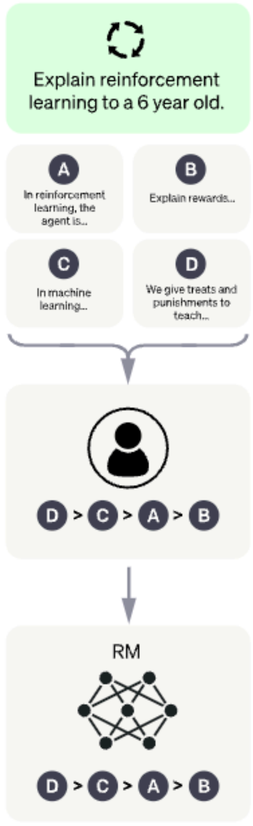
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

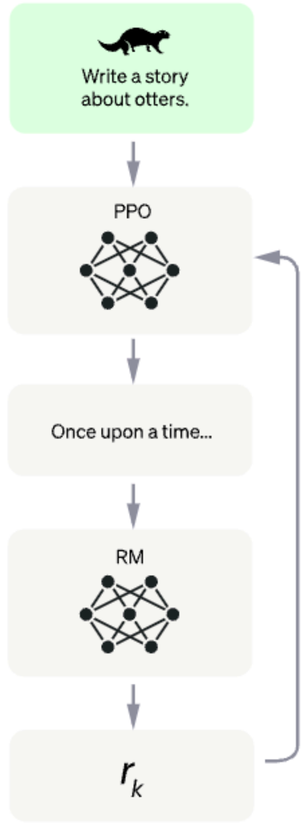
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Ouyang et al., (2022)

Collecting demonstration data

- Gather a large amount of task demonstrations
- Given an instruction, humans are tasked with annotating a **demonstration** of the task requested by the instruction
- Collected around 75k task demonstrations
  - 13k for supervised fine-tuning (SFT)
  - 31k to train the reward model (RM)
  - 33k examples to train with PPO (RLHF)

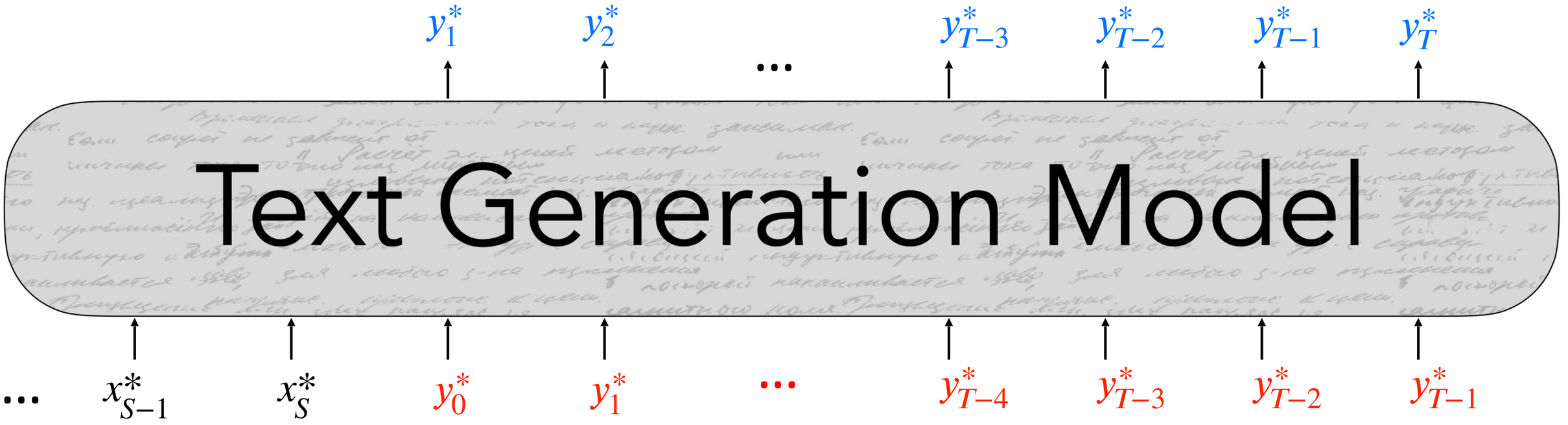
Ouyang et al., (2022)

How should we use this data we have now collected?

Step 1: Supervised fine-tuning (SFT)

- Trained to generate the next word of the demonstration y_t^* given a set of preceding words in demonstration $\{y^*\}_{<t}$ and instruction $\{x^*\}$

$$\mathscr{L} = -\sum_{t=1}^{T} \log P\left(y_t^* \mid \{y_{<t}^*\}; \{x^*\}\right)$$



What are shortcomings of only using SFT?

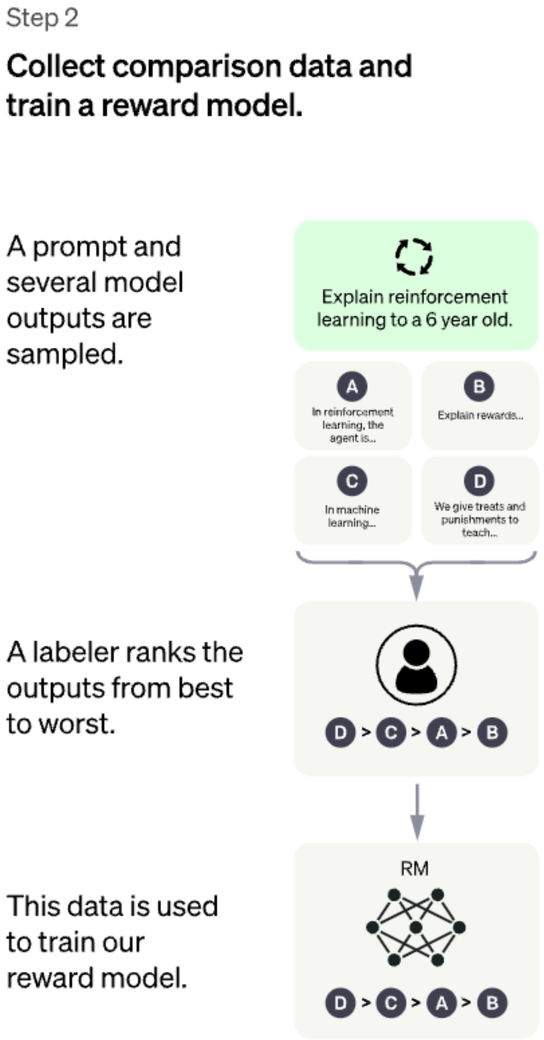
Expensive to collect lots data

Exposure bias

Plausibility vs. Factuality

What should we do instead?

Training a reward model



- Can't rely on human judgments to train our model for all demonstrations **(Expensive to collect)**
- Initialise reward model to predict a score for any demonstration that is provided as input
  - Score represents "quality" of demonstration given the instruction
- Train reward model to give higher scores
  - For two demonstrations Y_+ and $Y\_$ (you know one is better than other)

$$\mathscr{L}_{RM} = -\log \sigma\big(R(Y_+) - R(Y\_)\big)$$

Training a reward model

Train the reward model to give higher scores to "good" demonstrations

$$\mathcal{L}_{RM} = -\log \sigma\big(R(Y_+) - R(Y_-)\big)$$

Y_+ = preferred output (chosen by human)

Y_- = less preferred

- $R(Y_+) \gg R(Y_-) \rightarrow$ loss small
- $R(Y_+) \ll R(Y_-) \rightarrow$ loss ↑

As $R(Y_+) - R(Y_-) \rightarrow \infty$,
Then $\sigma(\,.\,) \rightarrow 1$,
And $\log(\,.\,) \rightarrow 0$

Now, for any demonstration Y we give to our reward model, we get a score $R(Y)$

Now that we have a reward model, how do we use it?

Naive approach:

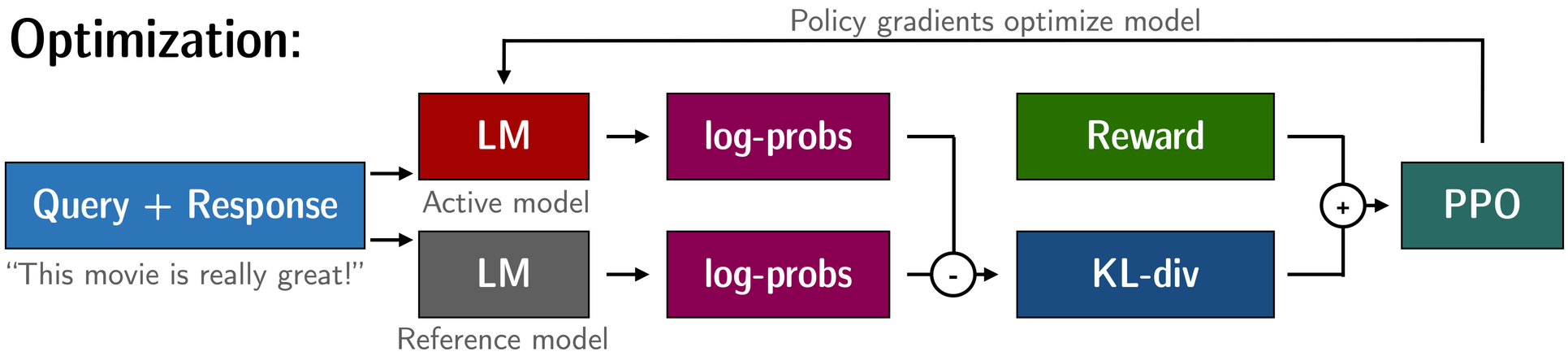
generate a bunch of demonstrations and use the highest scoring one

Optimising with Reinforcement Learning

Rollout:

Evaluation:

Optimization:

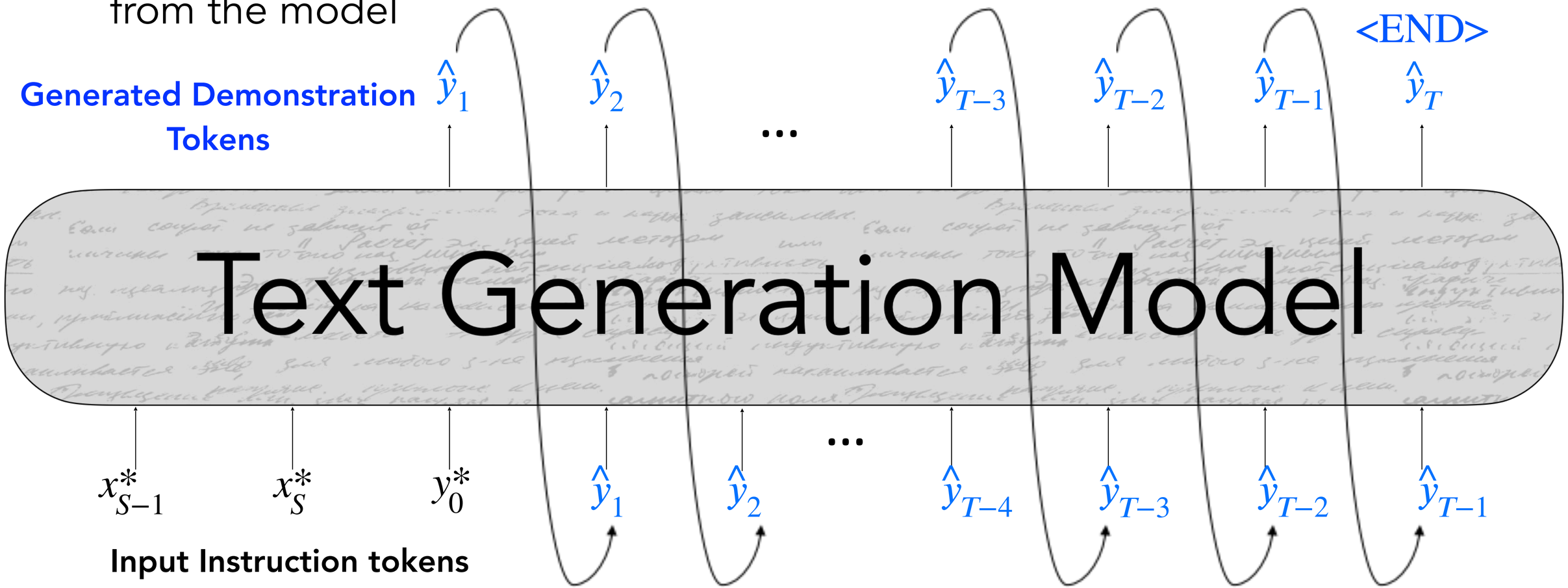


What reinforcement learning algorithm have we already seen?

REINFORCE

- Given **instruction,** sample **demonstration** from the model

$$\mathscr{L}_{RL} = -R(\hat{Y}) \sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)$$



REINFORCE: Basics

- Given **instruction,** sample **demonstration** from the model

$$\mathscr{L}_{RL} = -R(\hat{Y}) \sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)$$

Next time, increase the probability of this sampled token in the same context.

…but do it more if I get a high reward from the reward function.

Last week: Implementation Tricks

- Start from model pretrained with SFT
- Set appropriate baseline

$$\mathscr{L}_{RL} = -(R(\hat{Y}) - b) \sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)$$

- Have a term that continues to encourage coherence

$$\mathscr{L} = \mathscr{L}_{MLE} + \alpha \mathscr{L}_{RL}$$

REINFORCE vs. PPO

- REINFORCE

$$\mathscr{L}_{RL} = -(R(\hat{Y}) - b) \sum_{t=1}^{T} \log P\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)$$

- PPO

$$\mathscr{L}_{PPO} = -\min\left(R(\hat{Y}) \sum_{t=1}^{T} \log \frac{P_{curr}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}, g(\epsilon, R(\hat{Y}))\right)$$

$$g(\epsilon, R(\hat{Y})) = \begin{cases} (1+\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) \geq 0 \\ (1-\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) < 0 \end{cases}$$

How does PPO differ?

- PPO

$$\mathscr{L}_{PPO} = -\min\left(R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}, g(\epsilon, R(\hat{Y}))\right)$$

$$R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)} \qquad g(\epsilon, R(\hat{Y})) = \begin{cases} (1+\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) \geq 0 \\ (1-\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) < 0 \end{cases}$$

Extreme values when P_{curr} and P_{base} deviate

How does PPO differ?

- PPO

$$\mathscr{L}_{PPO} = -\min\left(R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}, g(\epsilon, R(\hat{Y}))\right)$$

$$R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)}$$

$$g(\epsilon, R(\hat{Y})) = \begin{cases} (1+\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) \geq 0 \\ (1-\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) < 0 \end{cases}$$

Min term here minimises how much your policy can deviate from the original base policy in every step

How does PPO differ?

- PPO

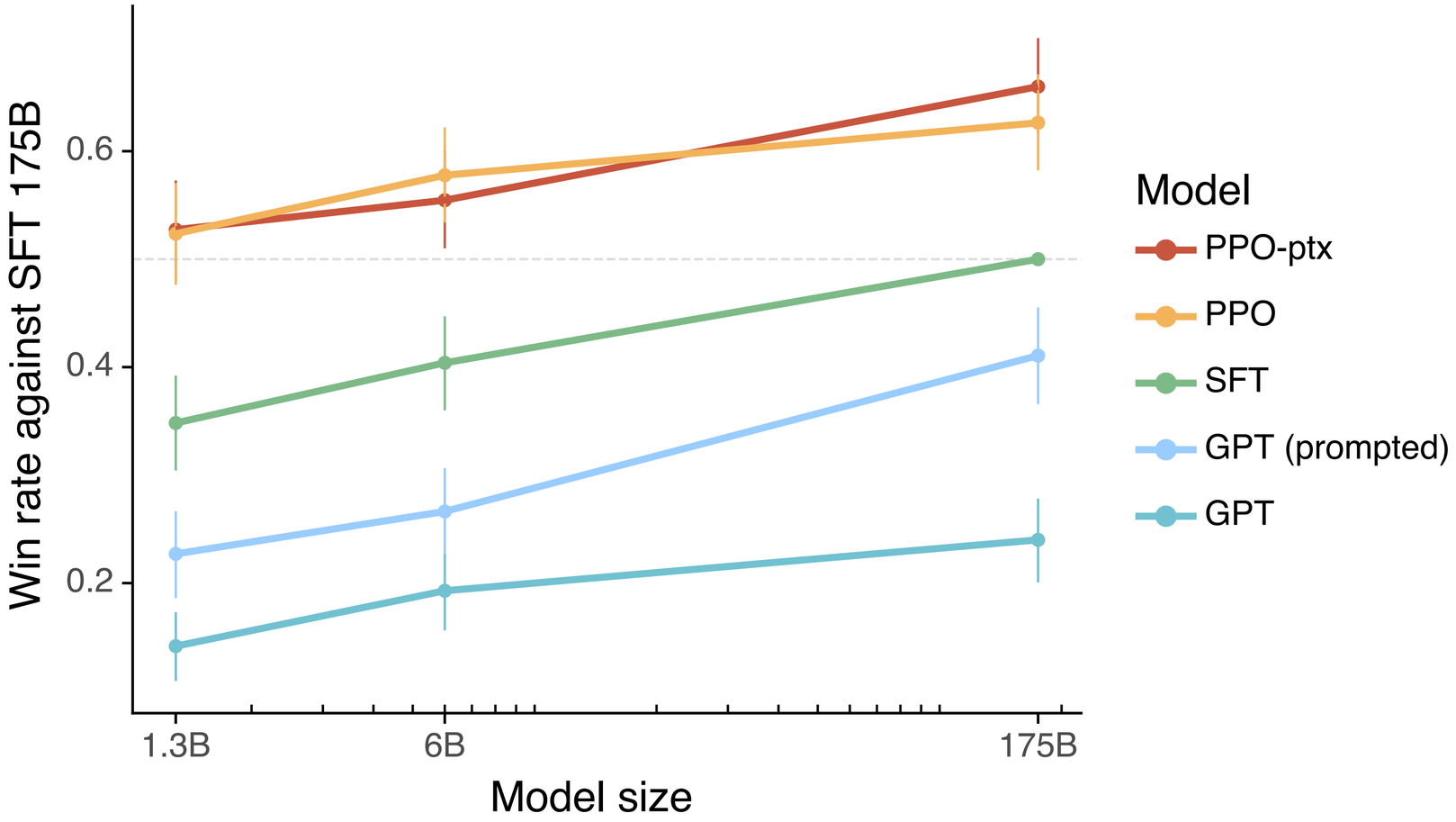
$$\mathscr{L}_{PPO} = -\min\left(R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{x^*\}; \{\hat{y}\}_{<t}\right)}, g(\epsilon, R(\hat{Y}))\right)$$

$$R(\hat{Y})\sum_{t=1}^{T}\log\frac{P_{curr}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)}{P_{base}\left(\hat{y}_t \mid \{y^*\}; \{\hat{y}\}_{<t}\right)}$$

$$g(\epsilon, R(\hat{Y})) = \begin{cases} (1+\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) \geq 0 \\ (1-\epsilon)R(\hat{Y}) & \textbf{if } R(\hat{Y}) < 0 \end{cases}$$

If reward is positive, loss is capped by $(1+\epsilon)R(\hat{Y})$

Outcome: Improvement



Instruction-tuned smaller models better at doing human tasks than GPT-3

Outcome: Many Tasks

AN

Can you give me a two-day itinerary for a weekend trip to Paris ? Please don't include more than three activities per day.

Day 1:

1. Visit the Eiffel Tower - Start your day with the iconic symbol of Paris, the Eiffel Tower. Take a stroll around the Champ de Mars and enjoy the views from the tower.
2. Explore the Louvre Museum - After lunch, head to the Louvre Museum and admire the world-famous artwork such as the Mona Lisa and the Venus de Milo.
3. Stroll along the Seine River - In the evening, take a romantic stroll along the Seine River and enjoy the Parisian atmosphere.

Day 2:

1. Visit Notre-Dame Cathedral - Start your day by visiting one of the most famous Gothic cathedrals in the world, Notre-Dame Cathedral.
2. Explore the Marais neighborhood - After lunch, explore the Marais neighborhood, known for its historical architecture, trendy boutiques, and restaurants.
3. Enjoy a Cabaret Show - In the evening, enjoy a cabaret show at the famous Parisian venue, the Moulin Rouge, and end your trip with a memorable experience.

AN

Can you generate the Python code for an attention mechanism ?

Here is an example of a simple attention mechanism implemented in Python:

Copy code

```python
import numpy as np

def attention(query, key, value, mask=None, dropout=None):
    "Compute 'Scaled Dot Product Attention'"
    d_k = query.shape[-1]
    scores = np.matmul(query, key.transpose()) / np.sqrt(d_k)

    if mask is not None:
        scores = scores - mask * 1e9

    scores = np.exp(scores)
    if dropout is not None:
        scores = dropout(scores)

    # Normalize the scores to weights
    weights = scores / np.sum(scores, axis=-1, keepdims=True)
    # Apply the weights to the values
    output = np.matmul(weights, value)

    return output, weights
```

This function computes the attention weights and outputs using the scaled dot product attention method, as described in the transformer architecture. It takes as input the query, key, and value vectors, as well as optional arguments for masking padded values and applying dropout to the attention scores.

Note that this is just a simple example, and there are many variations and improvements that can be made to the attention mechanism, such as using other methods for calculating the attention scores or incorporating more sophisticated masking schemes.

Outcome: Personalization

AN

Explain the causes of World War I in a three paragraph essay. Try to use as many complicated words as possible.

The causes of World War I can be traced back to several factors, including imperialism, nationalism, and the complex network of alliances between countries. Imperialism, or the pursuit of territorial expansion and political power, was a driving force behind the conflict as various nations sought to increase their influence and control over regions and resources. Nationalism, or a strong sense of pride and loyalty to one's country, was also a major contributing factor, as many people believed that their nation was superior to others and sought to assert its dominance.

The complex network of treaties and alliances between countries created an intricate web of obligations that ultimately led to the outbreak of war. The Triple Entente, comprising of France, Russia, and Britain, was formed in response to the formation of the Central Powers, including Germany, Austria-Hungary, and the Ottoman Empire. The system of alliances meant that, when one country was attacked, its allies were obligated to come to its aid, thereby drawing multiple nations into the conflict.

Additionally, the powder keg of Europe was primed for an explosion with the proliferation of military build-up, as nations sought to increase their military might. Tensions were further exacerbated by the assassination of Archduke Franz Ferdinand of Austria-Hungary by a member of the Serbian nationalist group, the Black Hand, in 1914. This event, known as the Sarajevo Incident, set off a chain of events that eventually led to the declaration of war. The complex interplay of these various factors ultimately culminated in the outbreak of World War I, a conflict that would shape the course of history for generations to come.

AN

Explain the causes of world war 1 to a 10 year old

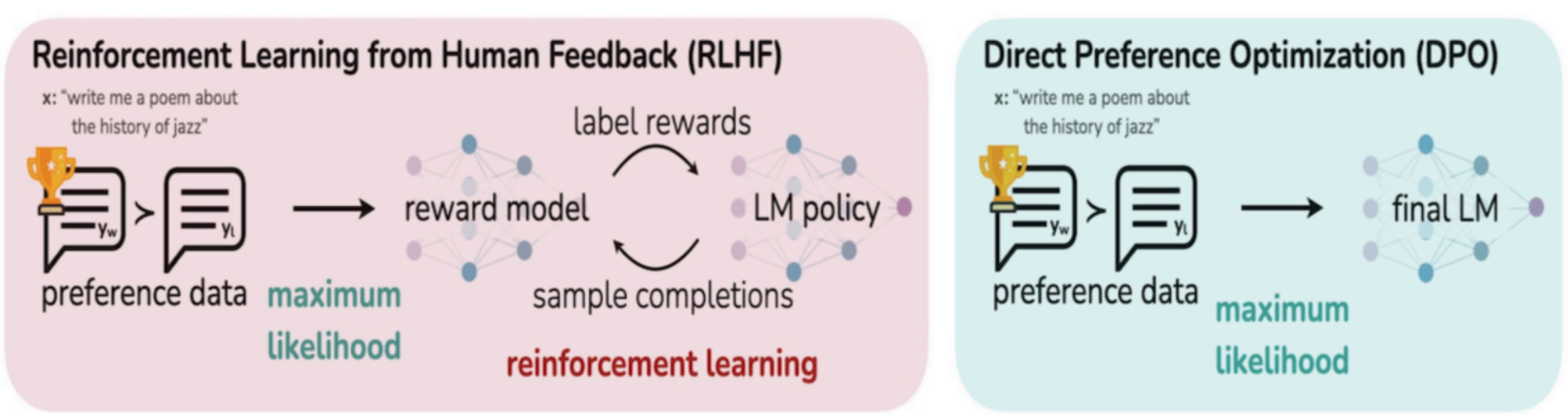
World War 1 was a big war that started in 1914 and lasted until 1918. It was fought between many countries, including Britain, France, and Russia on one side, and Germany, Austria-Hungary, and the Ottoman Empire on the other.

The main cause of World War 1 was a disagreement between countries over who was in charge of certain territories and the assassination of Archduke Franz Ferdinand of Austria-Hungary by a member of a Serbian nationalist group. This event led to a chain of events that eventually led to the start of the war.

Another factor that contributed to the start of the war was the military alliances between countries. Some countries had made agreements to support each other if one of them was attacked, which meant that if one country went to war, others would join too.

In short, World War 1 was caused by a combination of factors, including political and military tensions, territorial disputes, and alliances between countries that made it more likely for a small conflict to escalate into a large war.

Direct Preference Optimization



- Reward model in RLHF learns human preferences. In DPO, we optimize for human preferences directly:

$$\mathcal{L}_{DPO} = -\log \sigma\left(\beta \log \frac{P_{curr}(Y_+ | x^*)}{P_{base}(Y_+ | x^*)} - \beta \log \frac{P_{curr}(Y_- | x^*)}{P_{base}(Y_- | x^*)}\right)$$

Direct Preference Optimization

- Reward model in RLHF learns human preferences. In DPO, we optimize for human preferences directly

$$\mathcal{L}_{DPO} = -\log \sigma\left(\beta \log \frac{P_{curr}\left(Y_+ \mid x^*\right)}{P_{base}\left(Y_+ \mid x^*\right)} - \beta \log \frac{P_{curr}\left(Y_- \mid x^*\right)}{P_{base}\left(Y_- \mid x^*\right)}\right)$$

Learning objective encourages ratio of policies P_{curr} and P_{base} to be higher for preferred sample Y_+ than Y_-

Less flexible than RLHF in theory, but works well in practice

Recap

- Larger scale (more parameters, more data, more compute) leads to **emergent** abilities in language models
  - In-context learning
  - Chain-of-thought reasoning
- Emergent abilities enable new efficient methods to adapt models to downstream tasks
  - Prompt engineering
  - Prompt tuning
- Instruction tuning and RLHF make models adaptable to new, never-seen tasks as long as those tasks can be dictated using natural language