

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

Antoine Bosselut



Announcements

- **Assignment 3:** Due Sunday, 21/04/2024 at 11:59 PM
 - Office Hours: **today, 1 PM**; next week, Wednesday 1 PM
- **Assignment 1 Grades:** Released next week
- **Course Project:**
 - All ingredients covered by the end of class today
 - Description & discussion tomorrow!
- **Exercise Session:** Tomorrow, in-context learning

Assignment 1 Grading Review

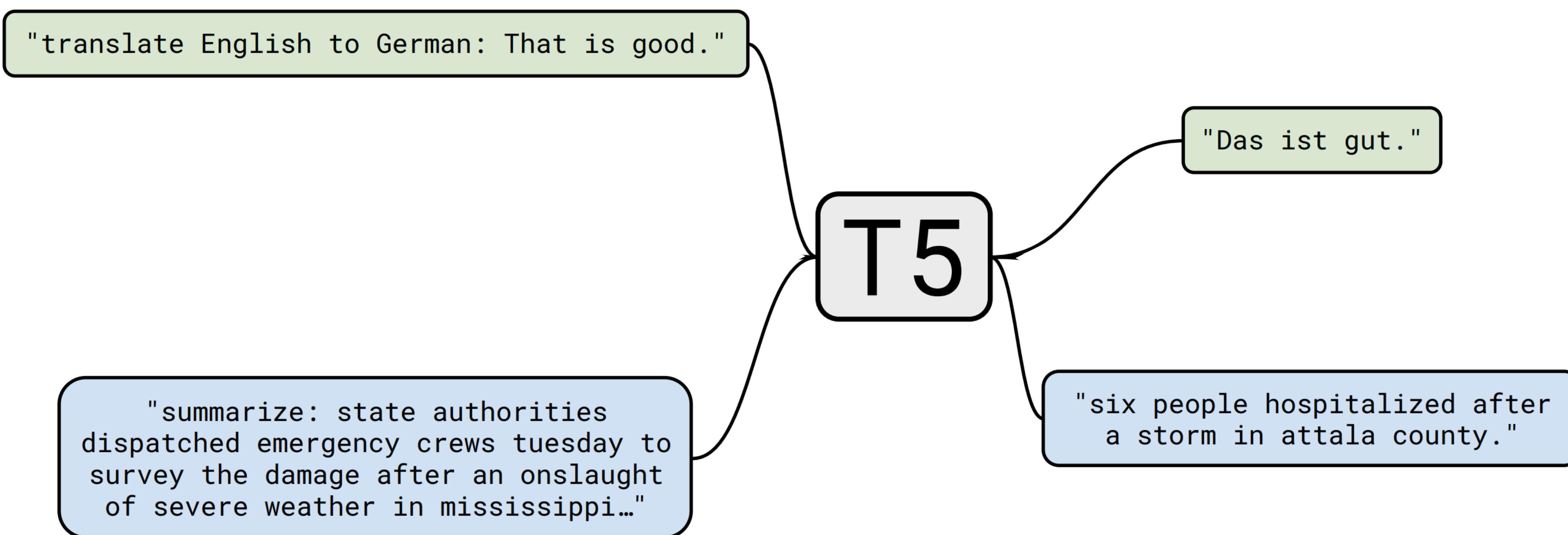
- **Assignment grades released early in the week (with feedback)**
 - Assignment rubric provided
 - Students review assignment grades, feedback, and rubric
- **Ed Discussion Threads created for each Assignment part**
 - Student make **private** messages about grading errors in relevant threads
 - TAs that graded relevant sections will engage with questions
- **Remaining grading disagreements to be discussed at Grade Review Sessions (April 18, 25)**
 - Students that started discussions on Ed will have priority
 - Ed messages made until **Wed., April 17th, noon** for priority on April 18th; **Wed. Apr 24th, noon** for priority in April 25th session
- **If disagreements remain, students can fill out Google Form for official re-grading**
 - Full assignment will be re-graded (it can go both ways!)

Today's Outline

- **Lecture**
 - **Quick Recap:** Fine-tuning, GPT-2
 - **GPT-3:** Scaling, Emergent Behavior, In-context Learning, Chain-of-Thought, ChatGPT
- **A3 Q&A Session**
- **Tomorrow:** Exercise Session
 - **Week 7 Exercise Session:** In-context learning
 - **Review of Week 6 Exercise Session:** Text Generation

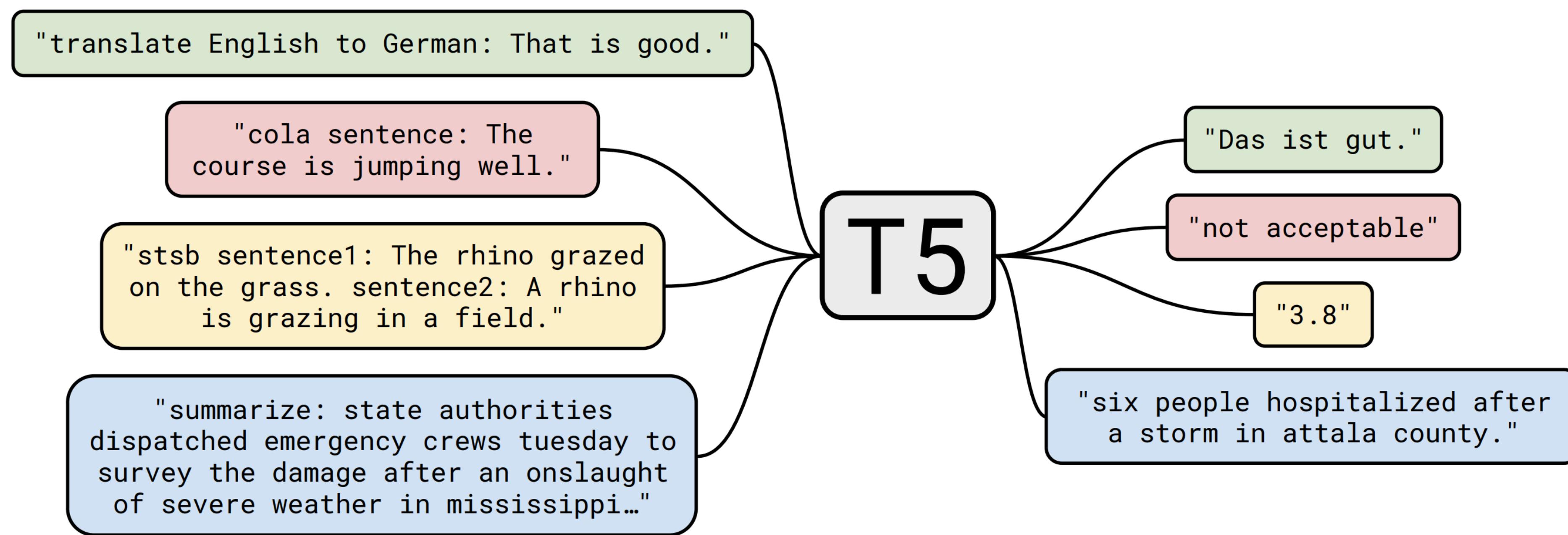
T5

Any problem can be cast as sequence-to-sequence



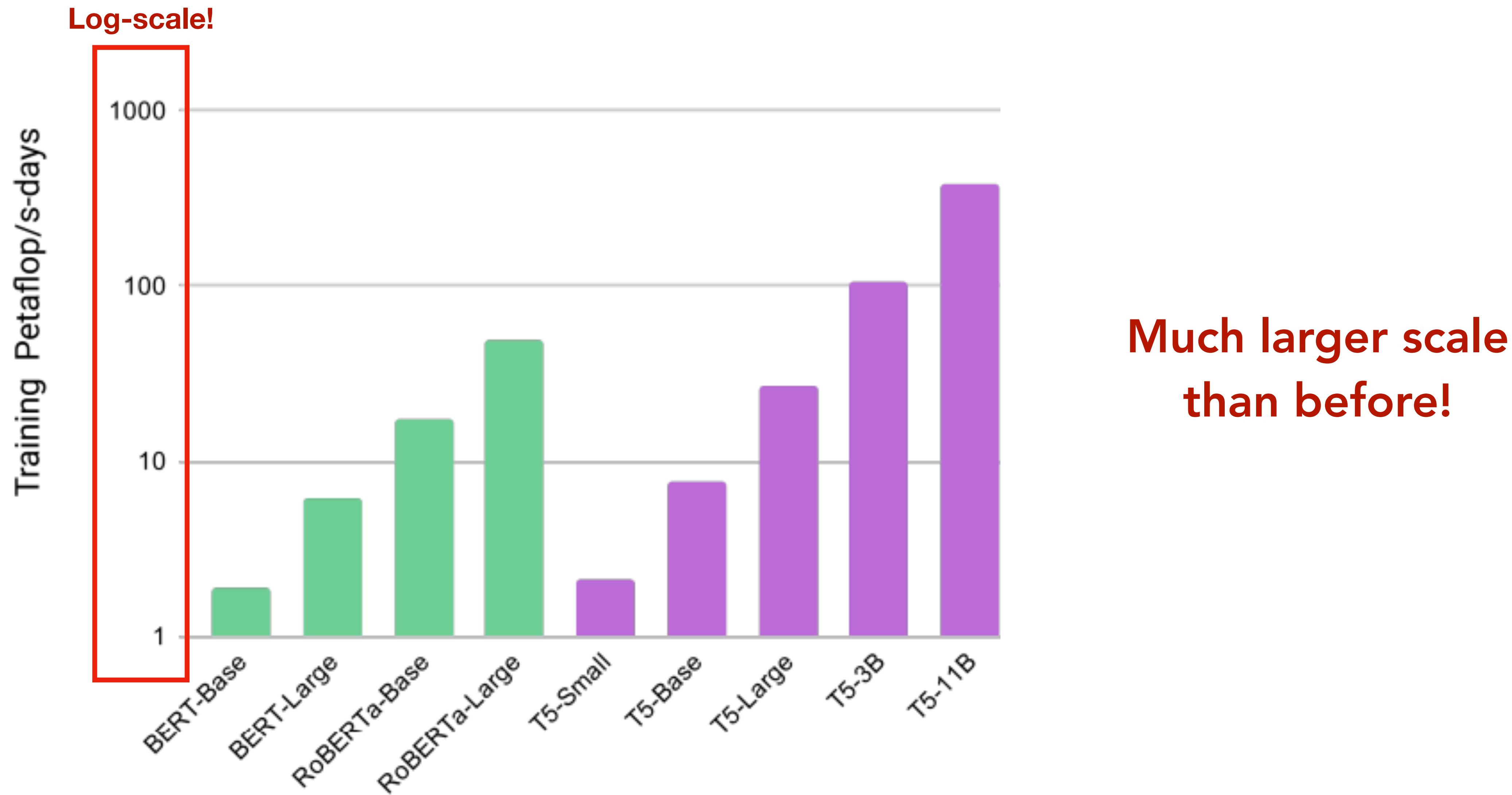
T5

Any problem can be cast as sequence-to-sequence

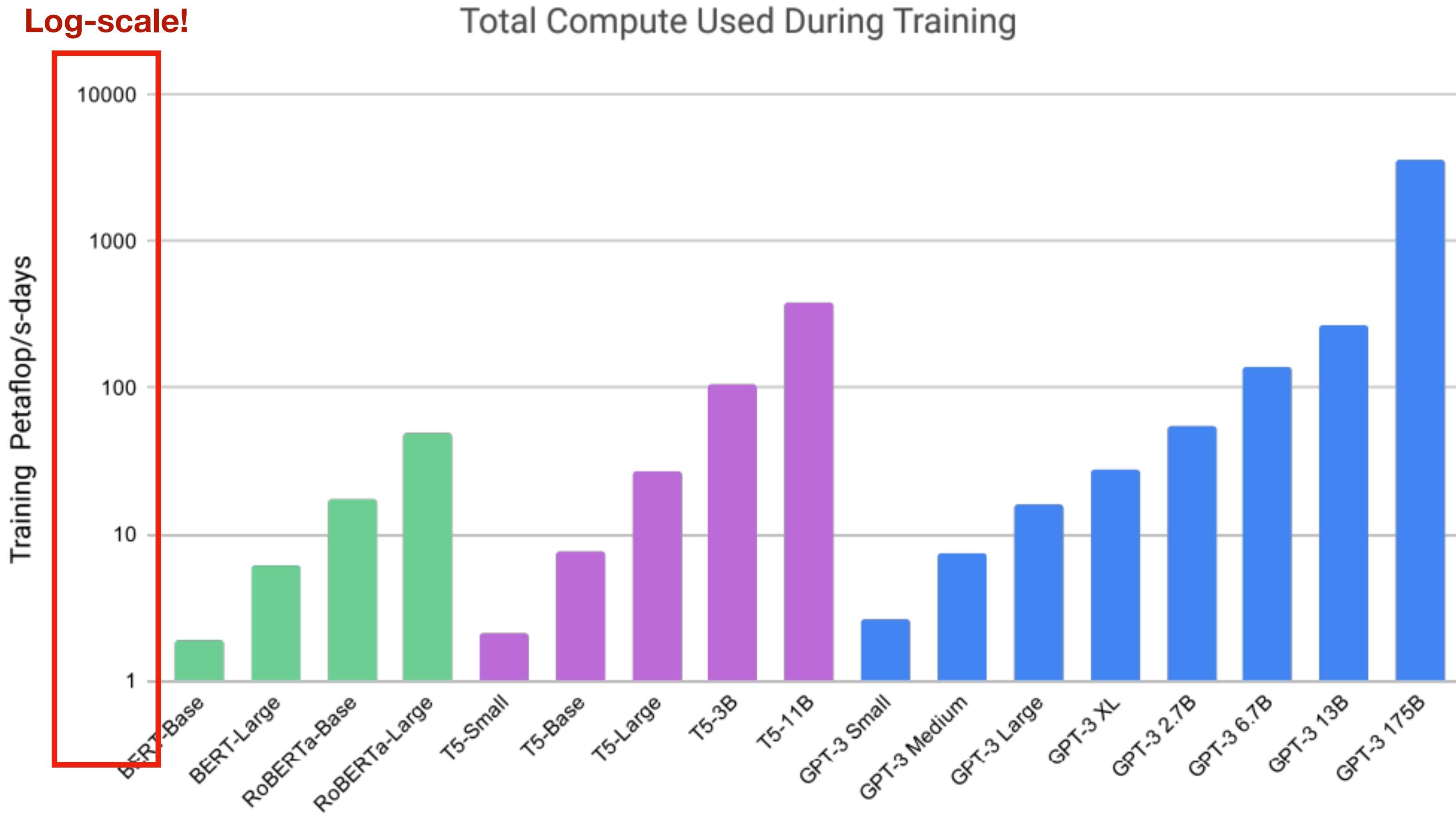


Rather than learning from the output embeddings of a [CLS] token, learn to generate the language that represents the semantics of the label

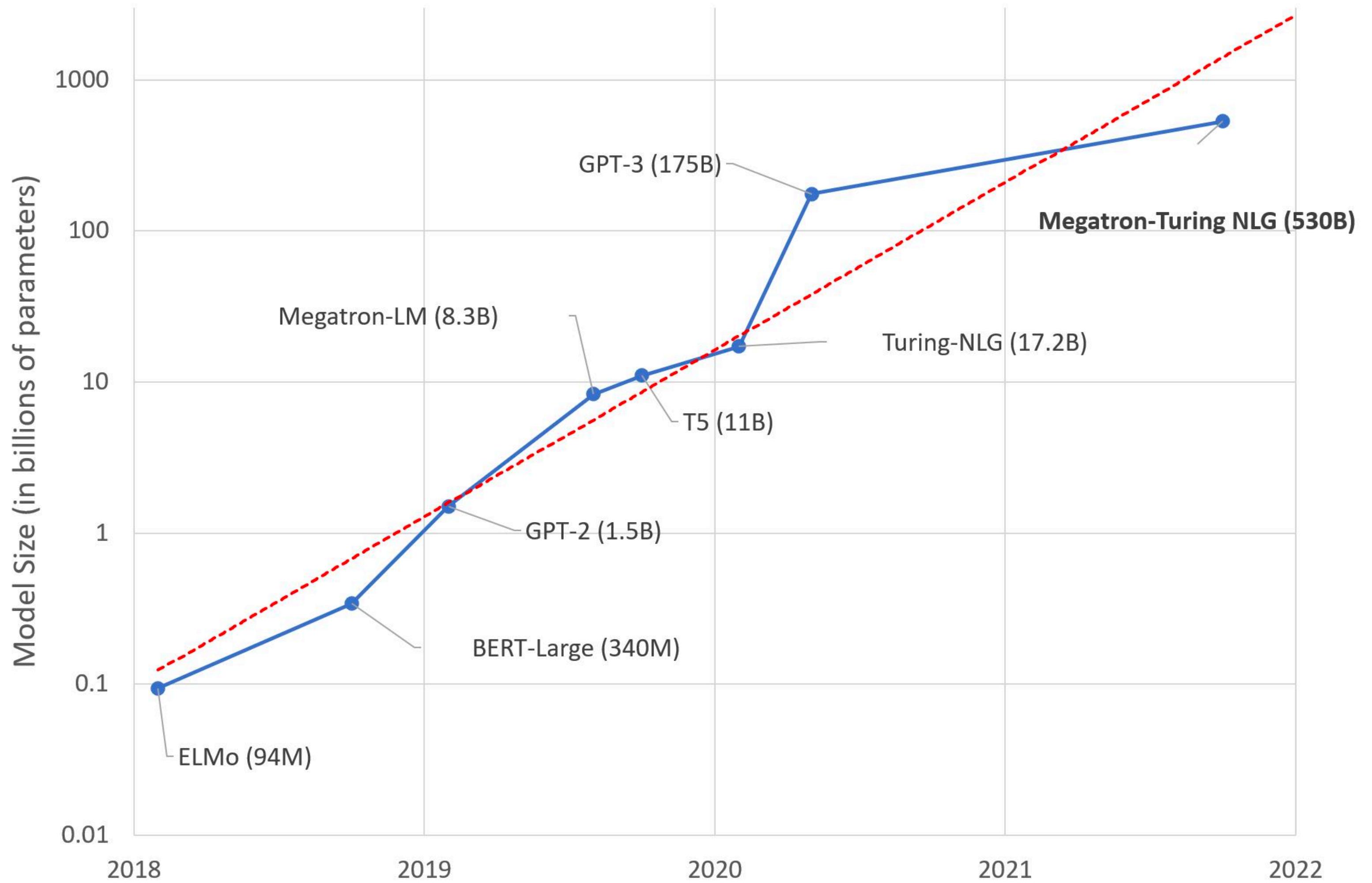
Language Model Scaling



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.”

- Philip Anderson, 1972

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

Controversial Topic!

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

Old Paradigm: Zero-shot (GPT2)

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

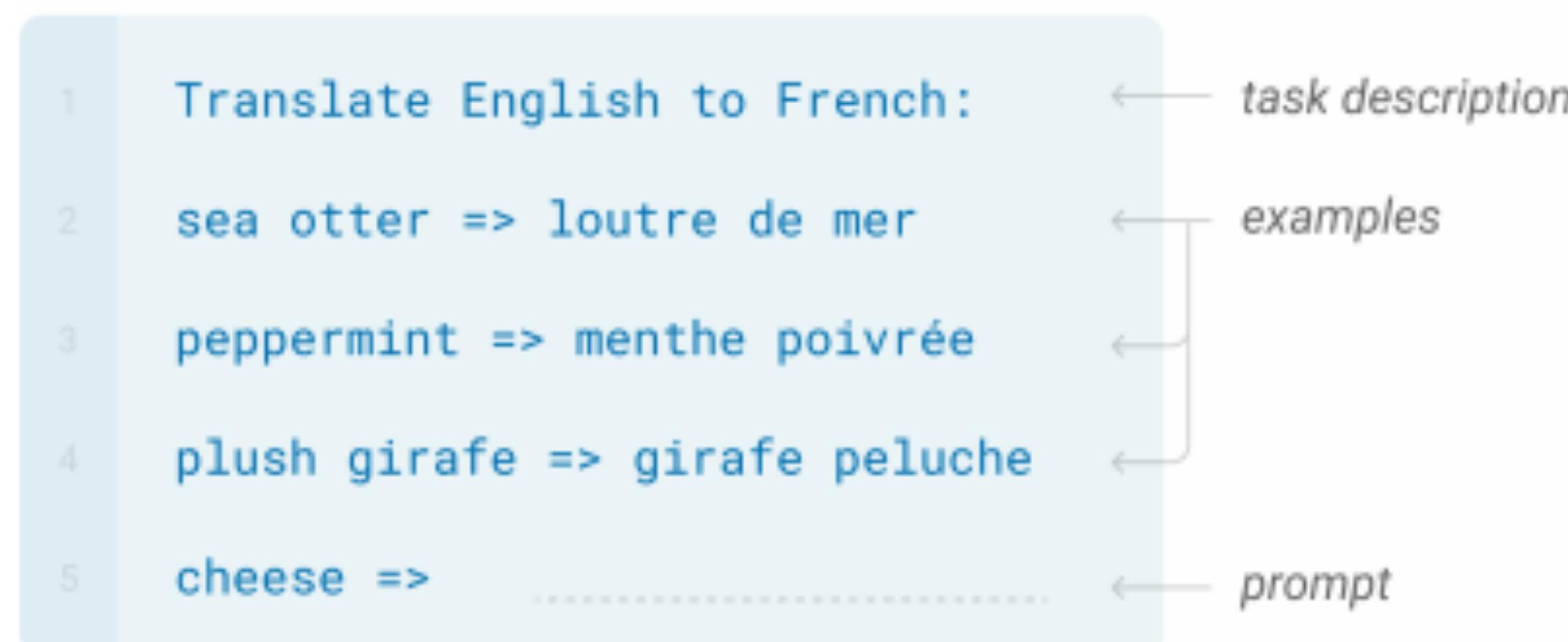


- In zero-shot adaptation, we use the pretrained model to complete tasks
 - The command we give the model to do a task: **task description**
 - The way we frame the example: **prompt**

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.

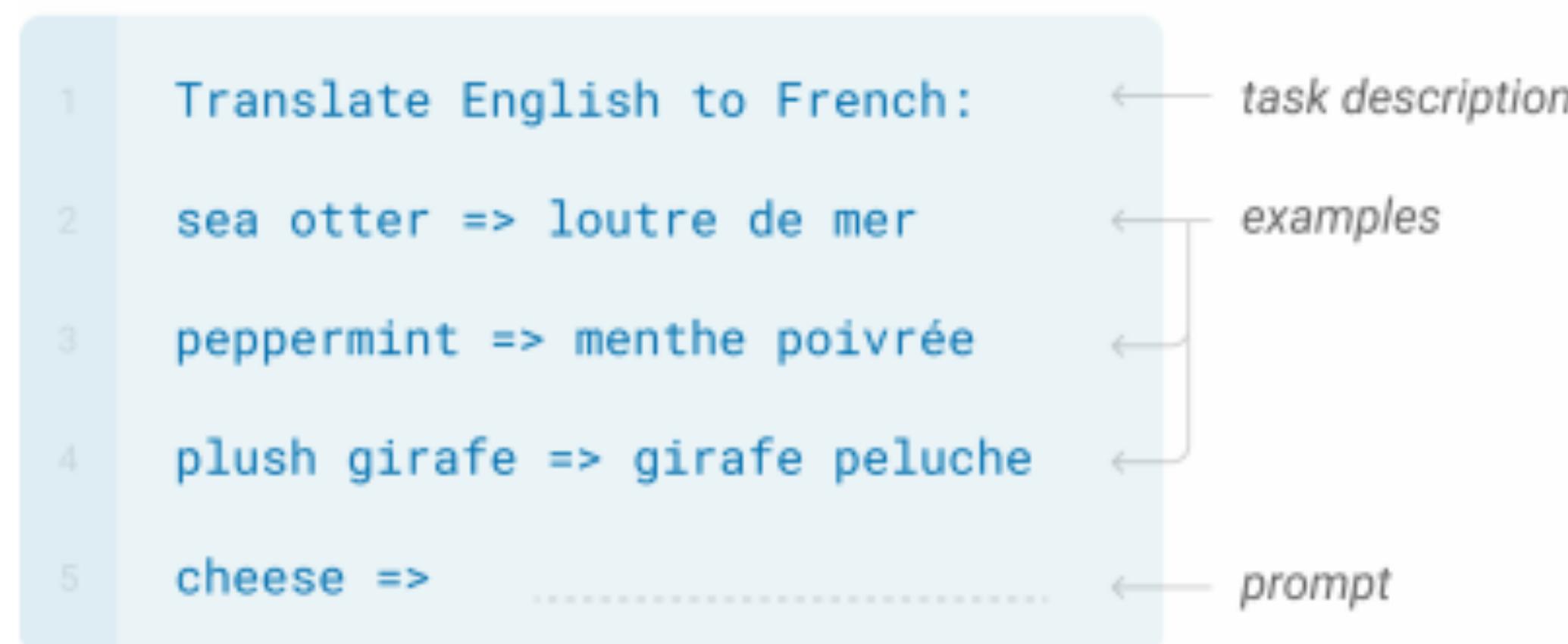


- In in-context learning, we concatenate the following in the **same** input sequence:
 - A task description (as in zero-shot learning)
 - An example (or multiple examples) of the task (i.e., context + label)
 - The final prompt, for which we want the model to predict a label

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.



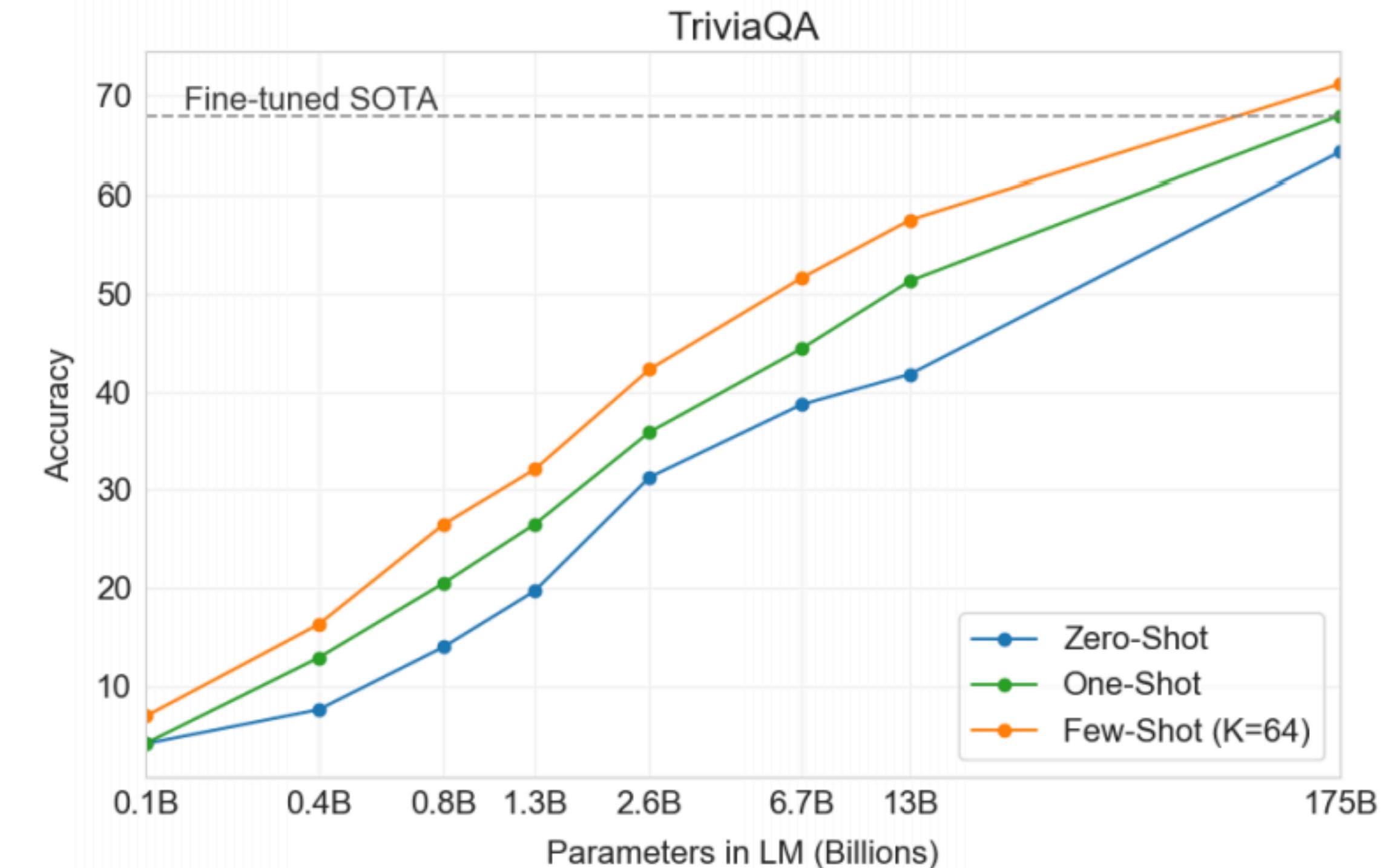
**There is no more learning
in the classical sense!**

- 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

TriviaQA Results

- Questions like:

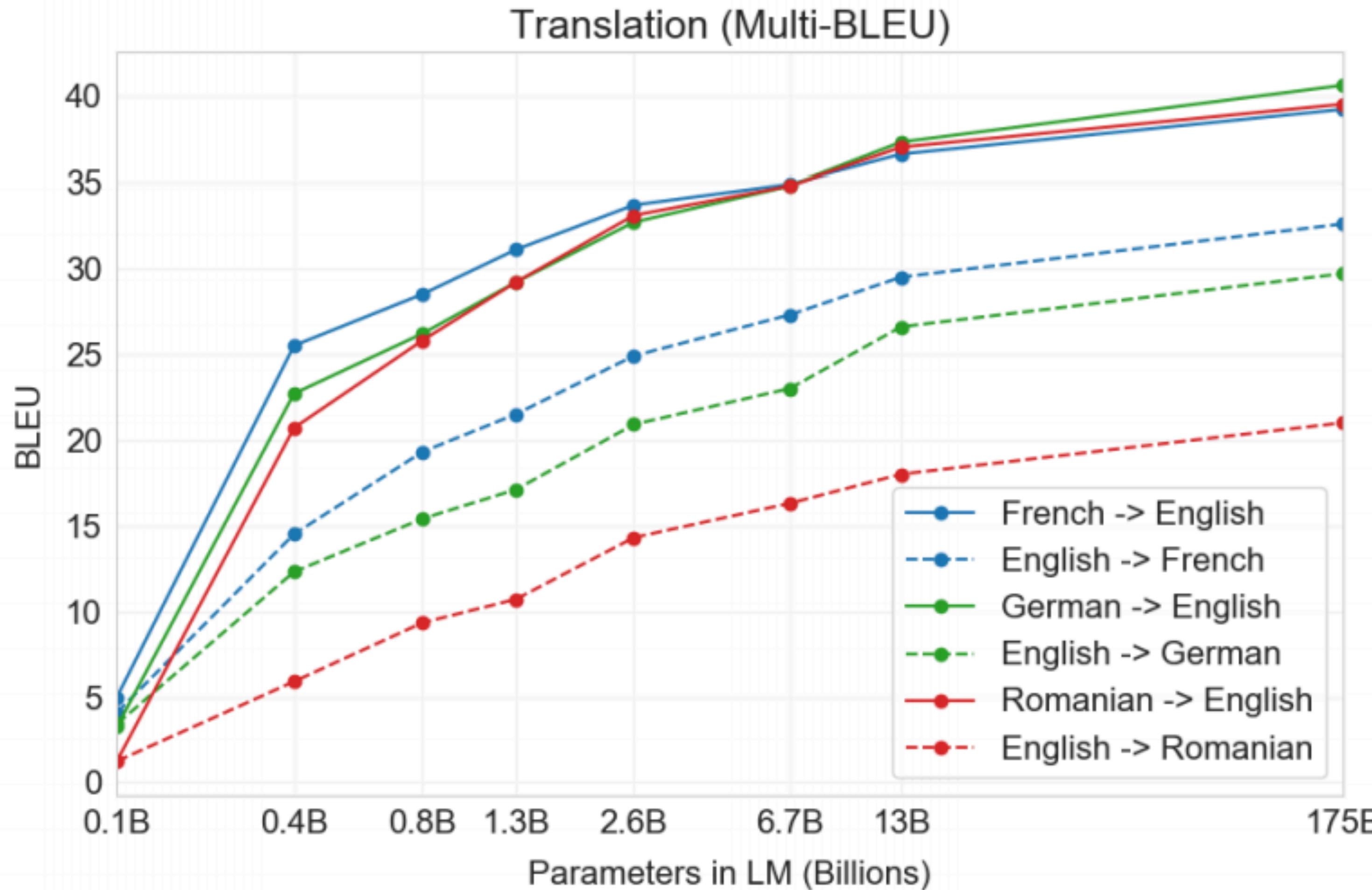
- Miami Beach in Florida borders which ocean?
- What was the occupation of Lovely Rita according to the song written by the Beatles?
- What was the Elephant man's real name?
- Which type of hat takes its name from an 1894 novel by George Du Maurier where the title character has the surname O'Farrell?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



No access to supporting documents!

Only memorisation!!!

Machine Translation



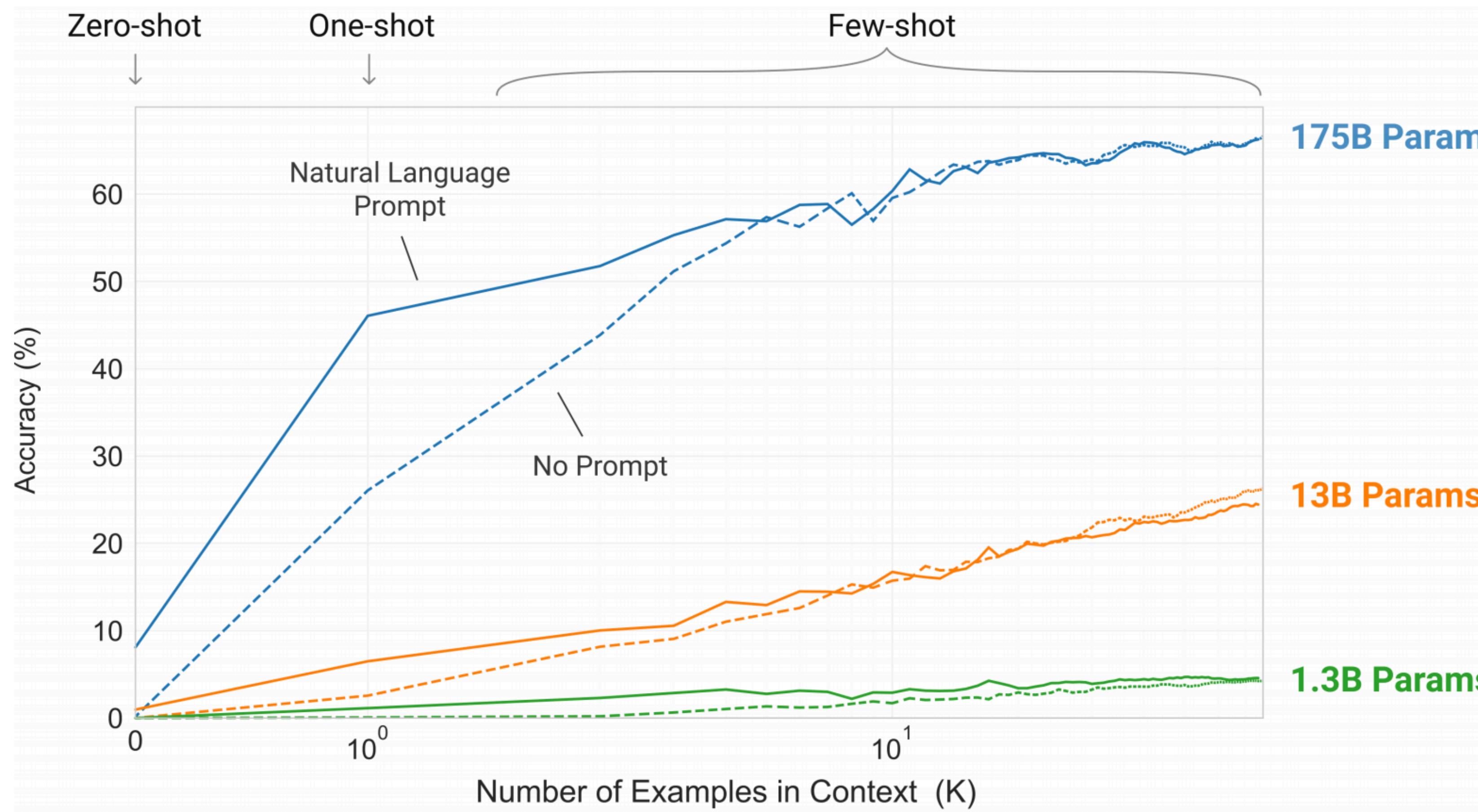
- Model not explicitly trained to translate from one language to another
- Model not **purposefully** trained on multilingual data
 - 7% of data is not English, though
- Model manages to learn **some** translation abilities

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]	<u>37.5</u>	34.9	28.3	35.2	<u>35.2</u>	33.1
mBART [LGG ⁺ 20]	-	-	<u>29.8</u>	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	<u>39.2</u>	29.7	<u>40.6</u>	21.0	<u>39.5</u>

Model much better at translating into English

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!

In-context Learning is sensitive

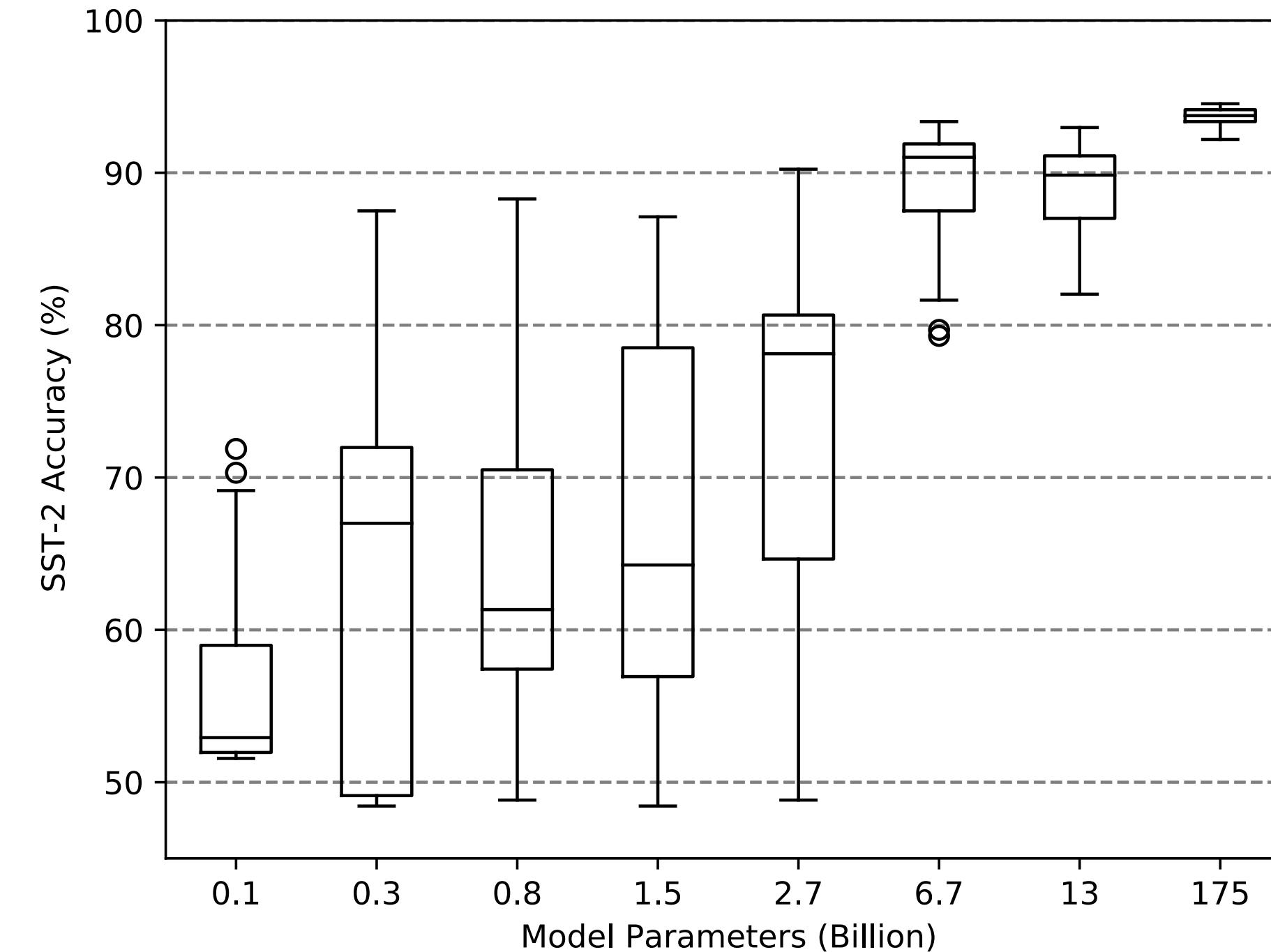
Trial	1	2	3	4	5
Accuracy	94.6	95.0	95.8	93.9	86.9

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!

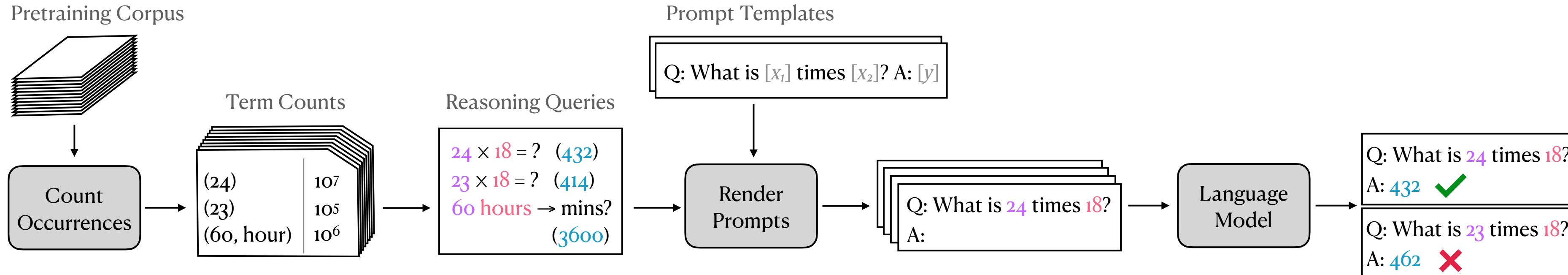
Performance distribution across 24 random orders of in-context examples



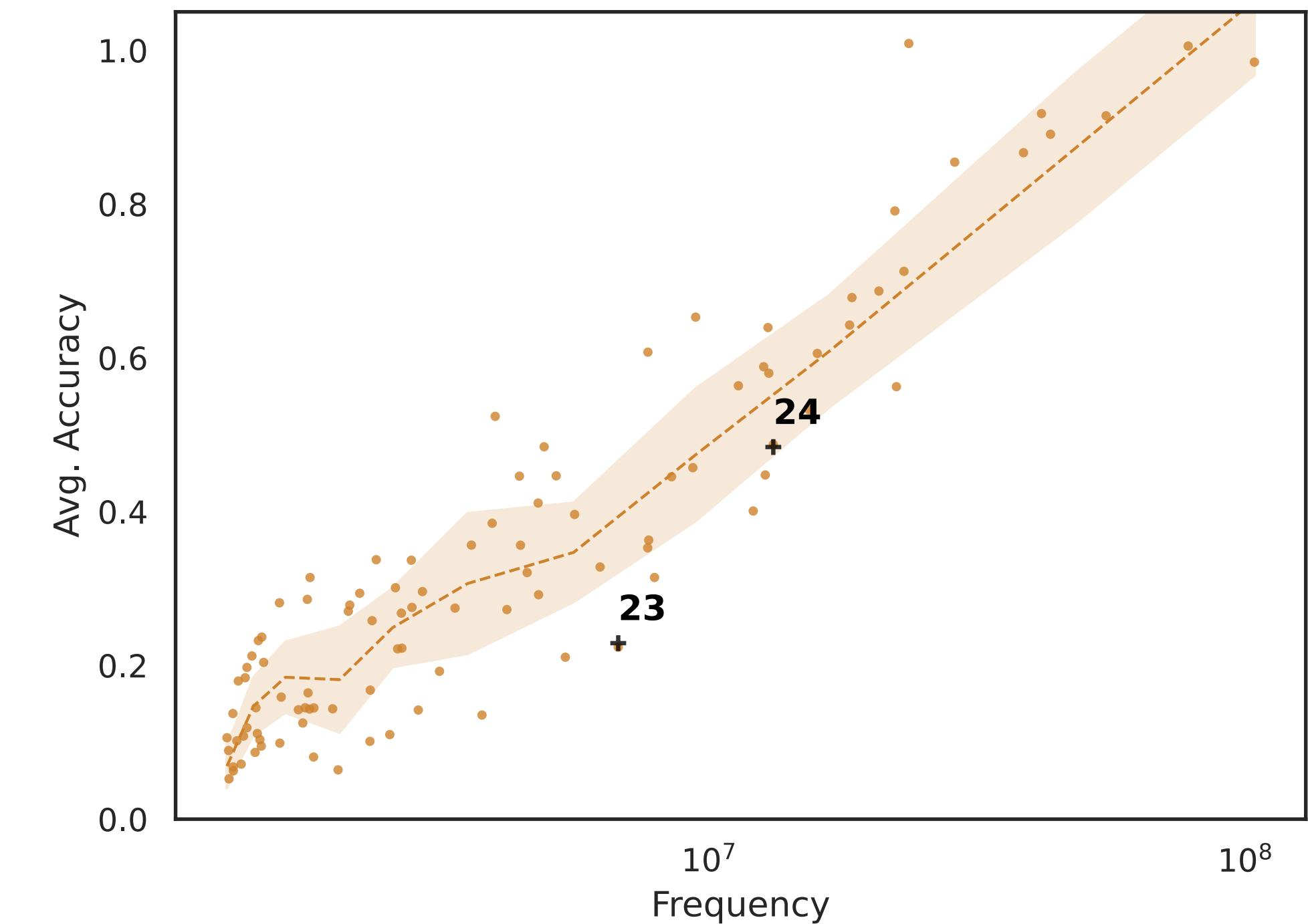
Lu et al., (2021)

Large variance in performance.
Example order matters!

Why does in-context learning work ?



- 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 :

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

$$P_3(a) = a. \text{ All in all, it was _____.}$$

$$P_4(a) = a \| \text{In summary, the restaurant is _____.}$$

We define a single verbalizer v for all patterns as

$$\begin{array}{lll} v(1) = \text{terrible} & v(2) = \text{bad} & v(3) = \text{okay} \\ v(4) = \text{good} & v(5) = \text{great} \end{array}$$

“verbalizer” of actual labels

“patterns” for expressing labels

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 :

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

$$P_3(a) = a. \text{ All in all, it was _____.}$$

$$P_4(a) = a \| \text{In summary, the restaurant is _____.}$$

“patterns” for expressing labels

We define a single verbalizer v for all patterns as

$$\begin{array}{lll} v(1) = \text{terrible} & v(2) = \text{bad} & v(3) = \text{okay} \\ v(4) = \text{good} & v(5) = \text{great} \end{array}$$

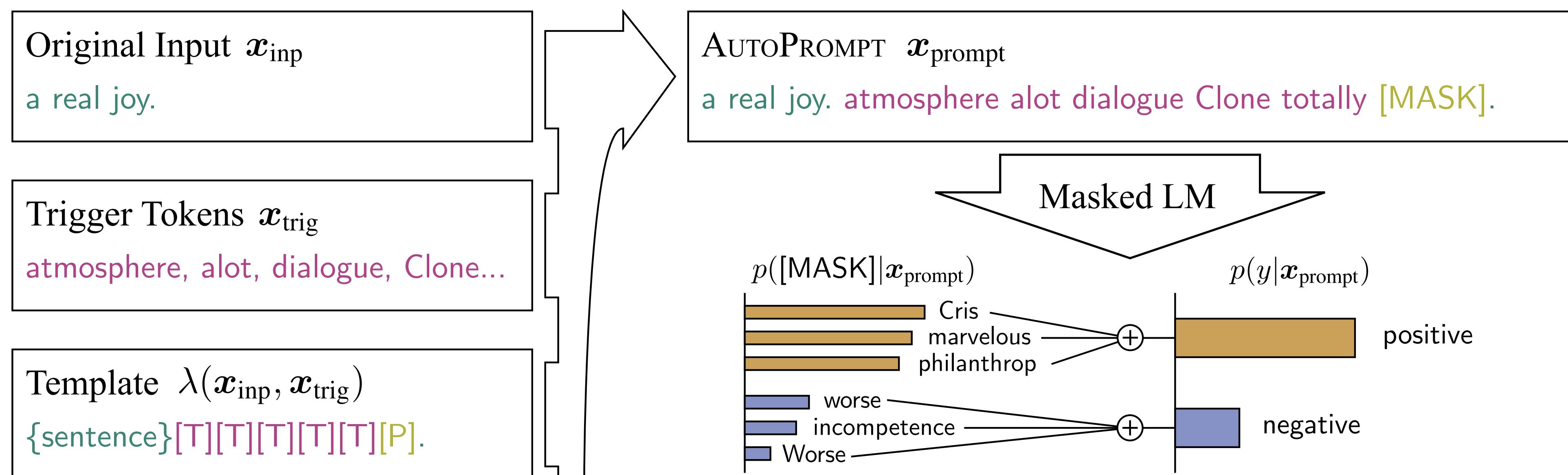
“verbalizer” of actual labels

- 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!

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



What makes for the best prompt?

Task	Prompt Template	Prompt found by AUTO PROMPT	Label Tokens
Sentiment Analysis	{sentence} [T]...[T] [P].	unflinchingly bleak and desperate Writing academics where 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	<i>X plays Y music</i> {sub}[T]...[T][P].	Hall Overton fireplacemade antique son alto [MASK].	
Relation Extraction	<i>X is a Y by profession</i> {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”

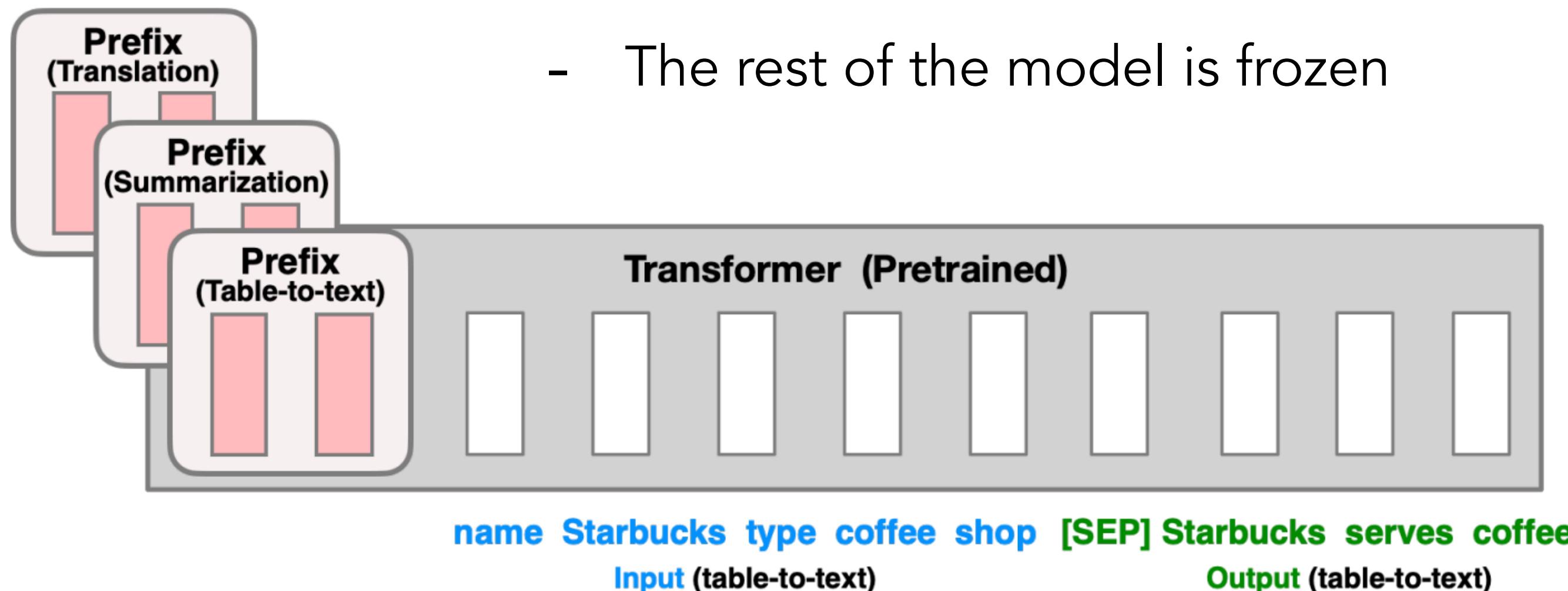
Optimal prompt search yields random sequences of tokens (i.e., vectors)

What else could we do?

Learn the prompt!

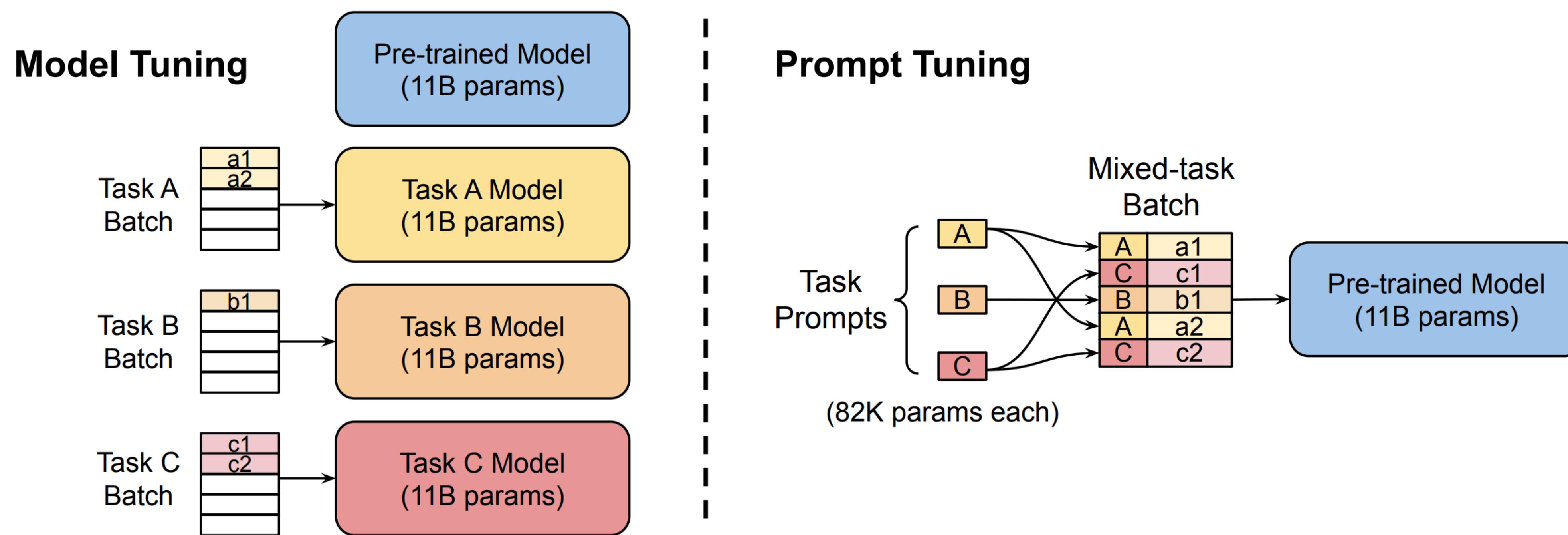
“Soft prompts” or Prompt Tuning

- 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



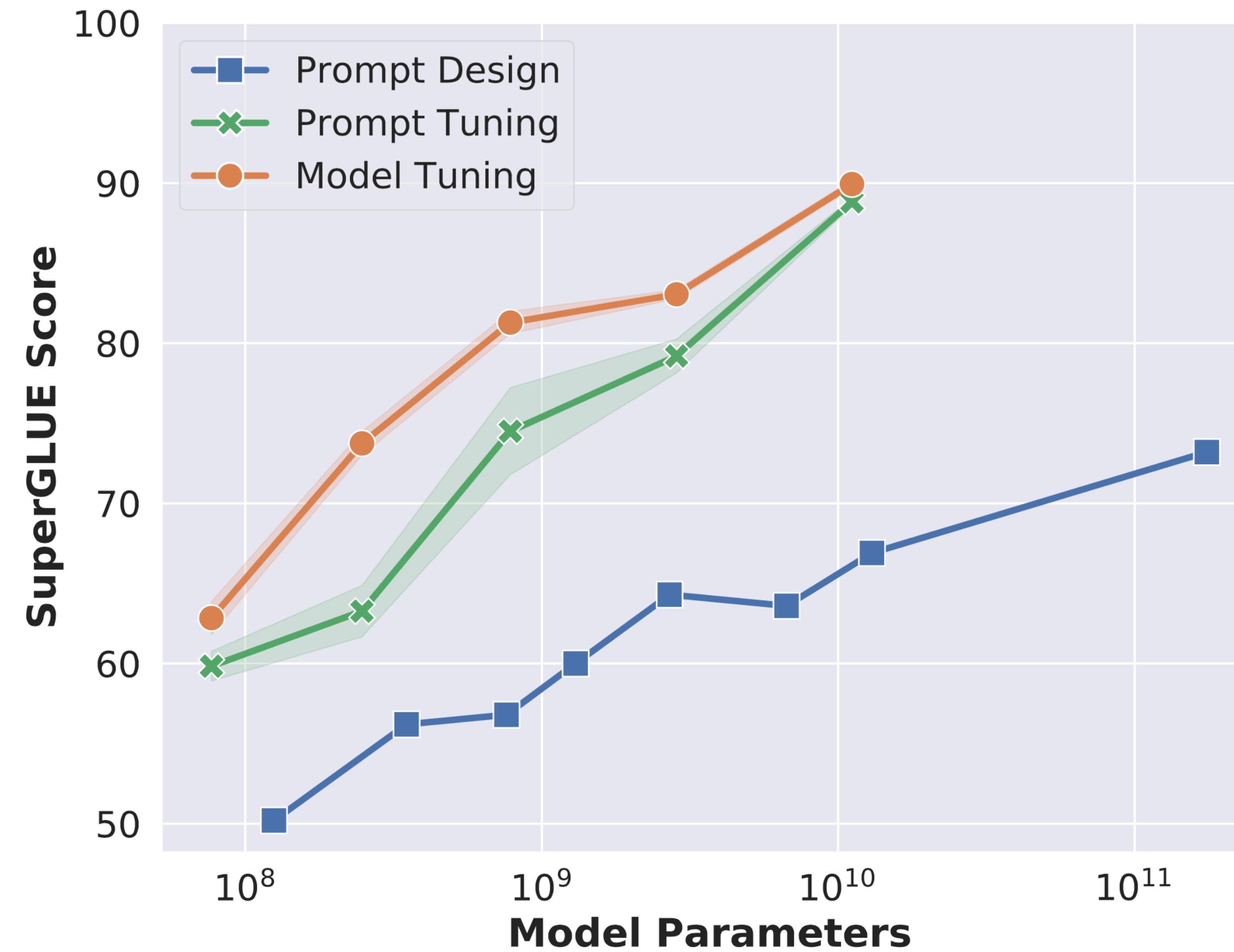
What might be some
(dis)advantages of prompt tuning?

“Soft prompts” or Prompt Tuning



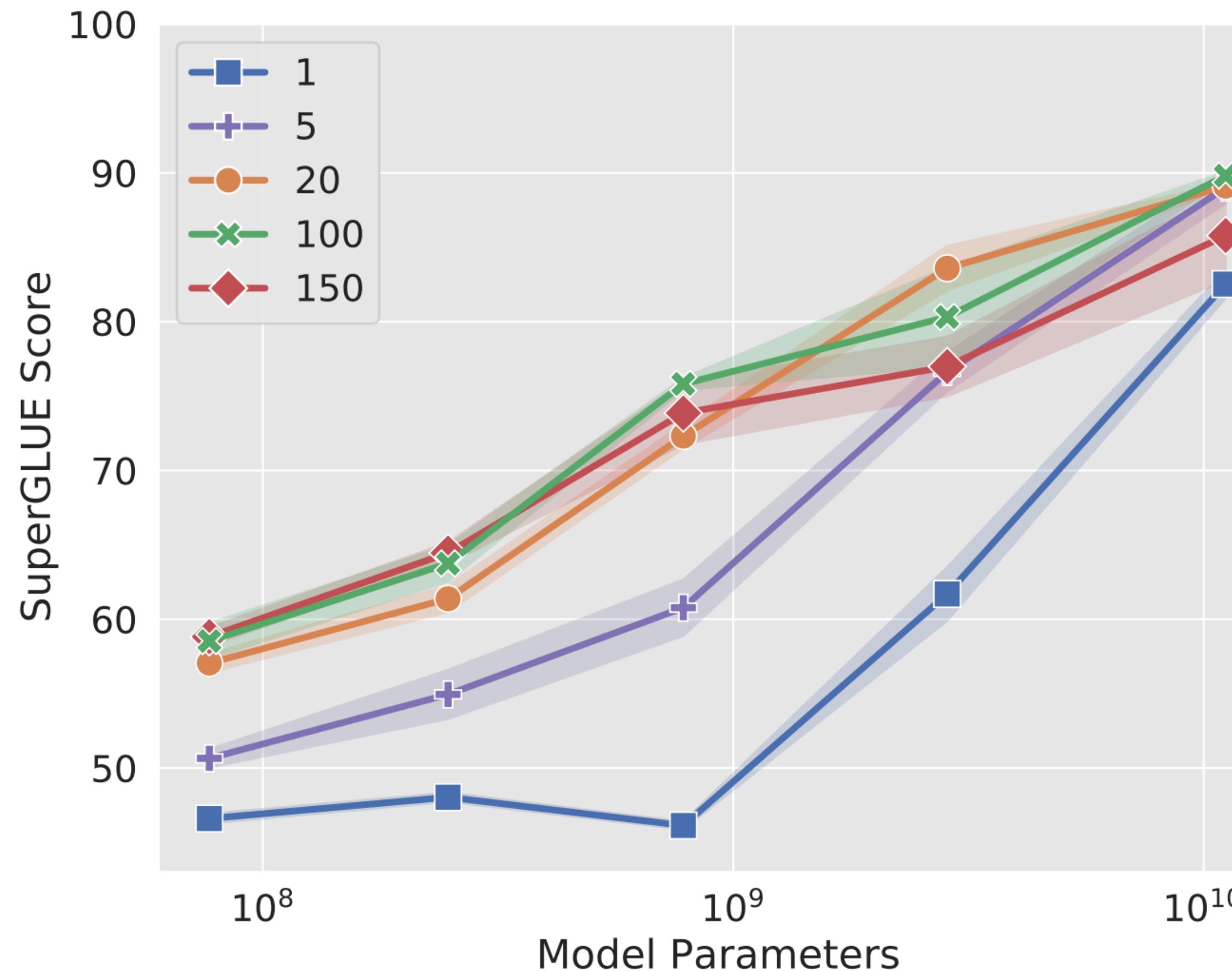
- Fine-tuning: update a new version of the model for each task
- Prompt tuning: update the prompt vectors for each task. **Other parameters are frozen.**
 - More efficient training (and separable multi-task learning)
 - Not as interpretable as discrete prompts!

Prompt Tuning



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

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

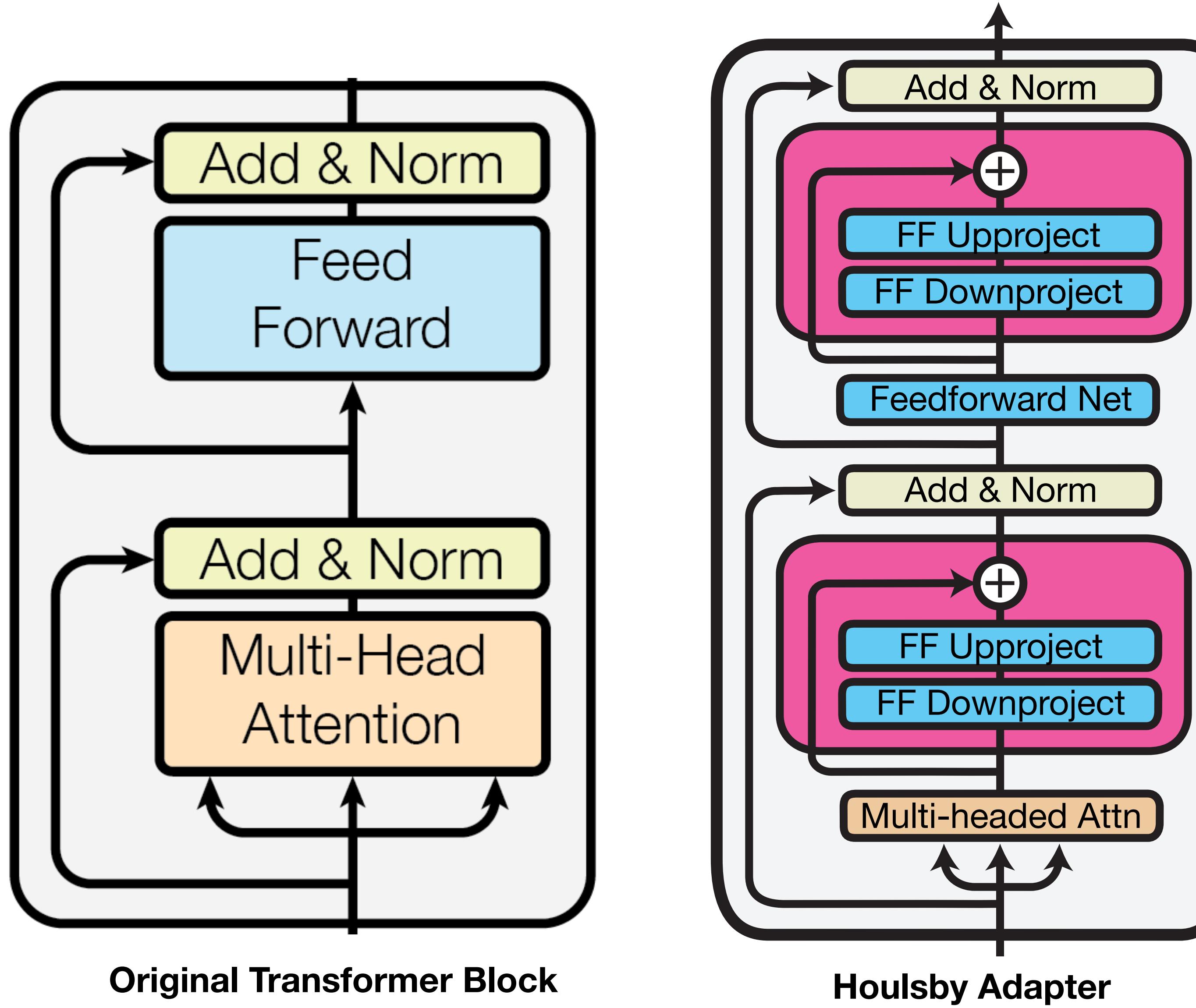
Prompt Tuning



- 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

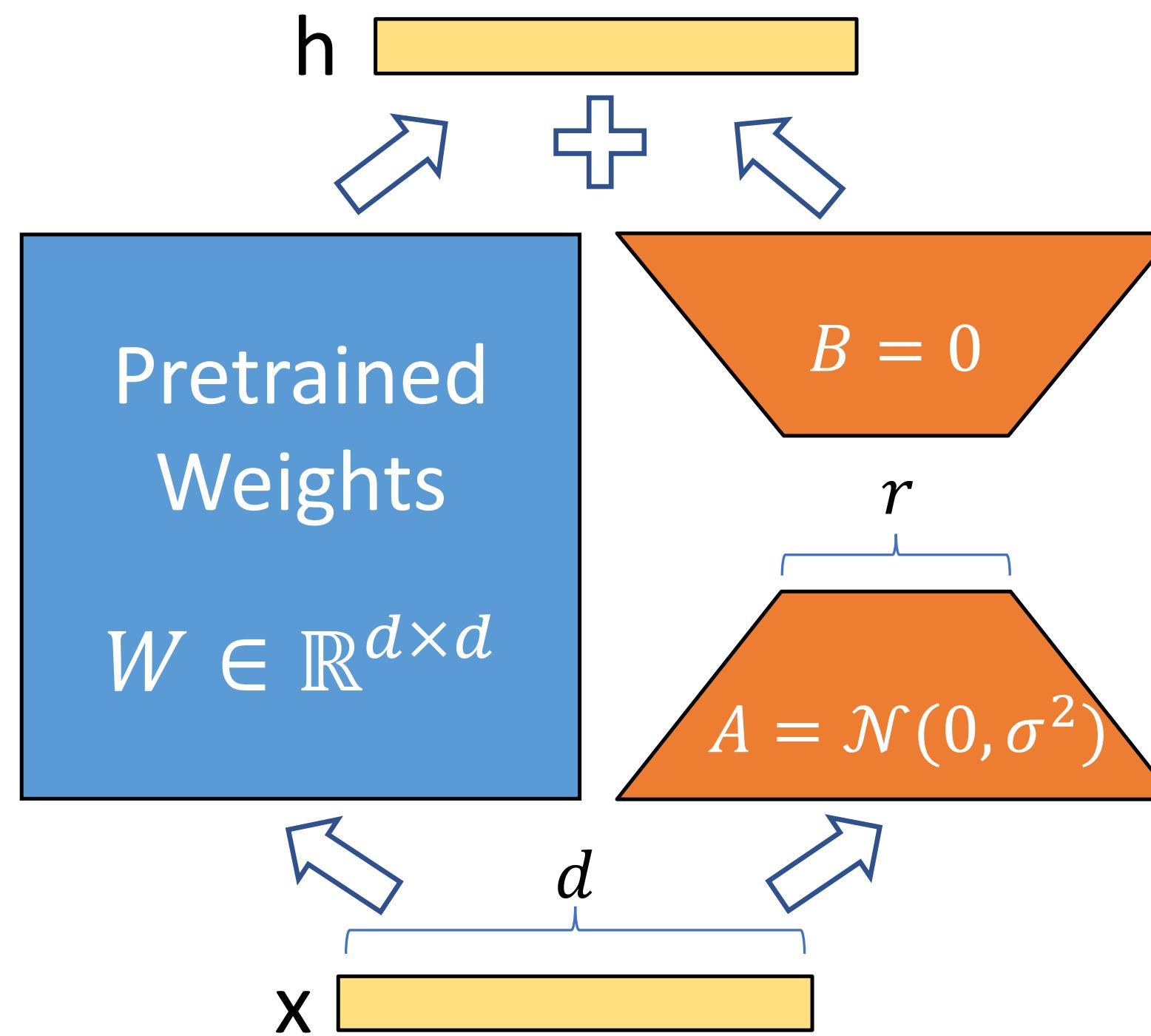
**Need large models for
these benefits to emerge!**

Efficient tuning: Adapters



- 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**

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**

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. 

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. X

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. ✓

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?

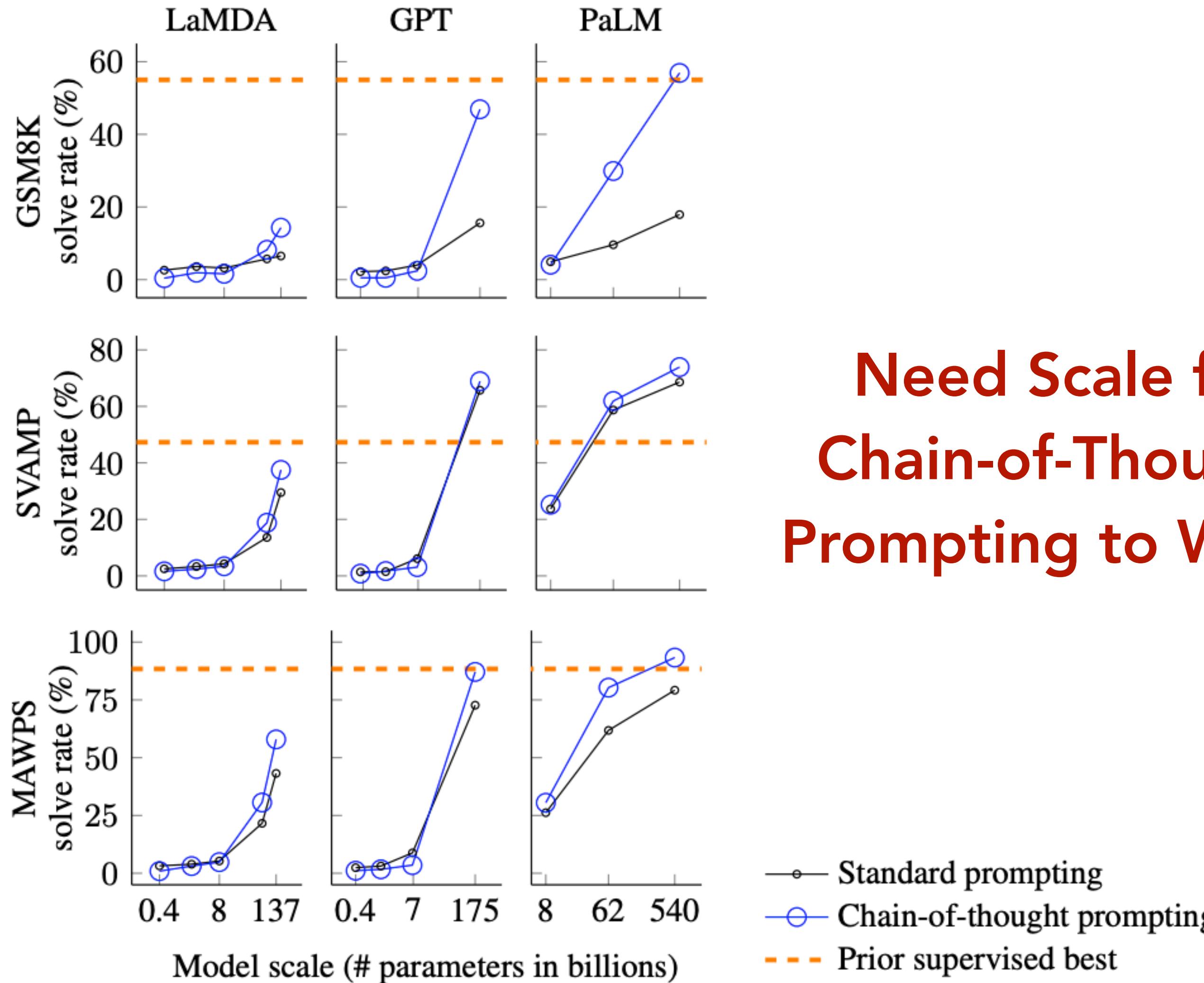
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

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. ✓

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. ✓

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.

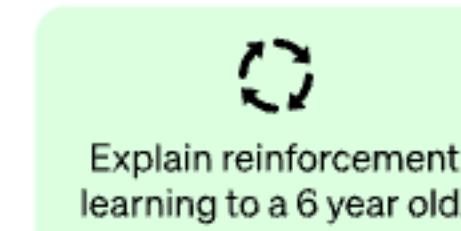
These models can understand and respond to a wide range of requests, including providing explanations for their responses. However, it is important to note that AI language models like GPT-4 are not perfect and can sometimes generate incorrect or incomplete information. As AI continues to evolve, newer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

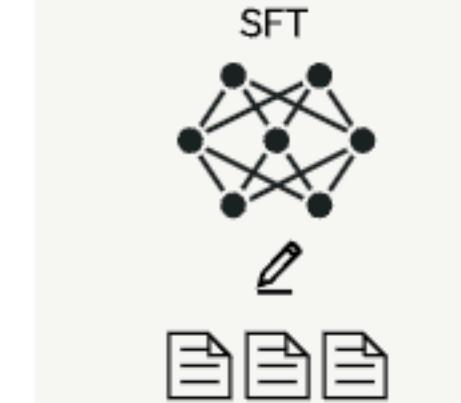
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



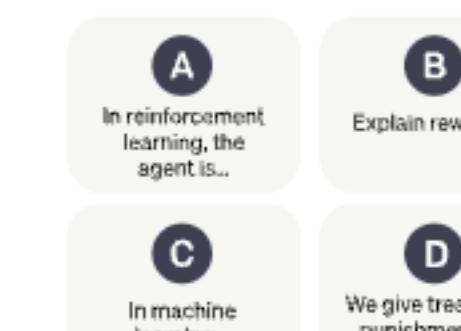
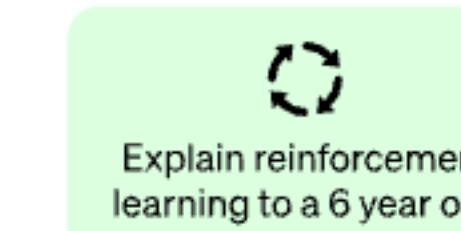
This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

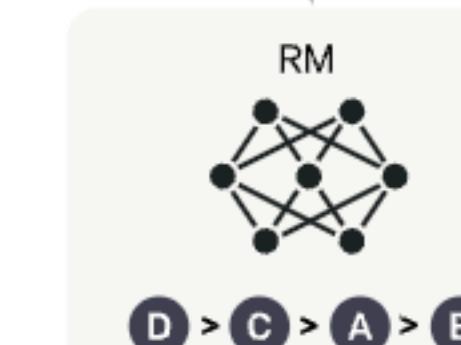
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



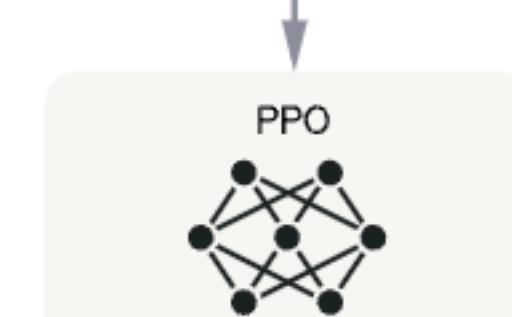
Step 3

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

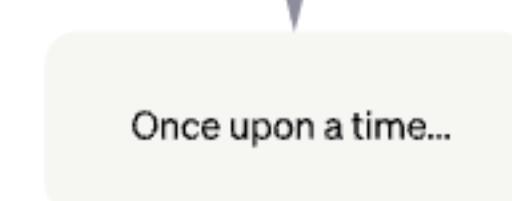
A new prompt is sampled from the dataset.



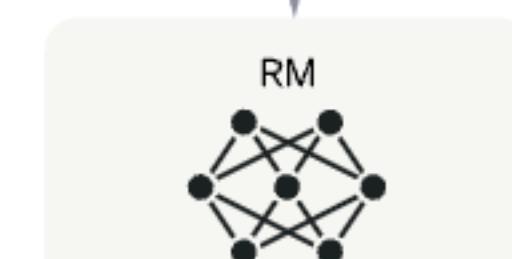
The PPO model is initialized from the supervised policy.



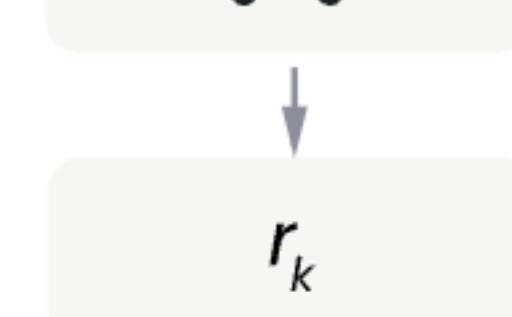
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Collecting demonstration data

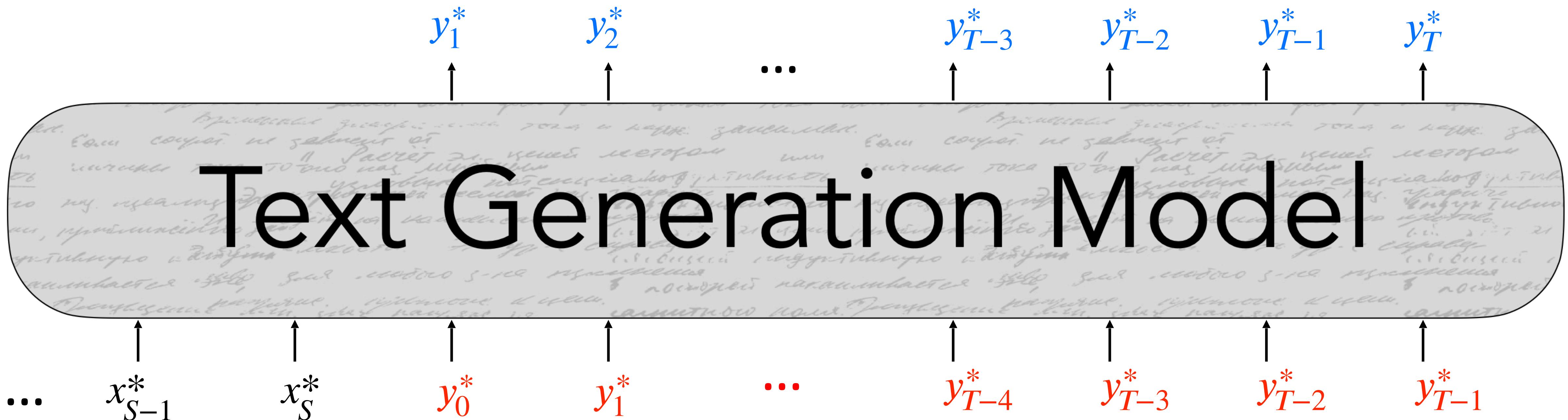
- Gather a large amount of task demonstrations
- Given an instruction, humans are tasked with annotating a **demonstration** of the task requested by the instruction
- Collected around 75k task demonstrations
 - 13k for supervised fine-tuning (SFT)
 - 31k to train the reward model (RM)
 - 33k examples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

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 $\{x^*\}$

$$\mathcal{L} = - \sum_{t=1}^T \log P(y_t^* | \{y_{<t}^*\}; \{x^*\})$$



What are shortcomings of only using SFT?

Expensive to collect lots data

Exposure bias

Plausibility vs. Factuality

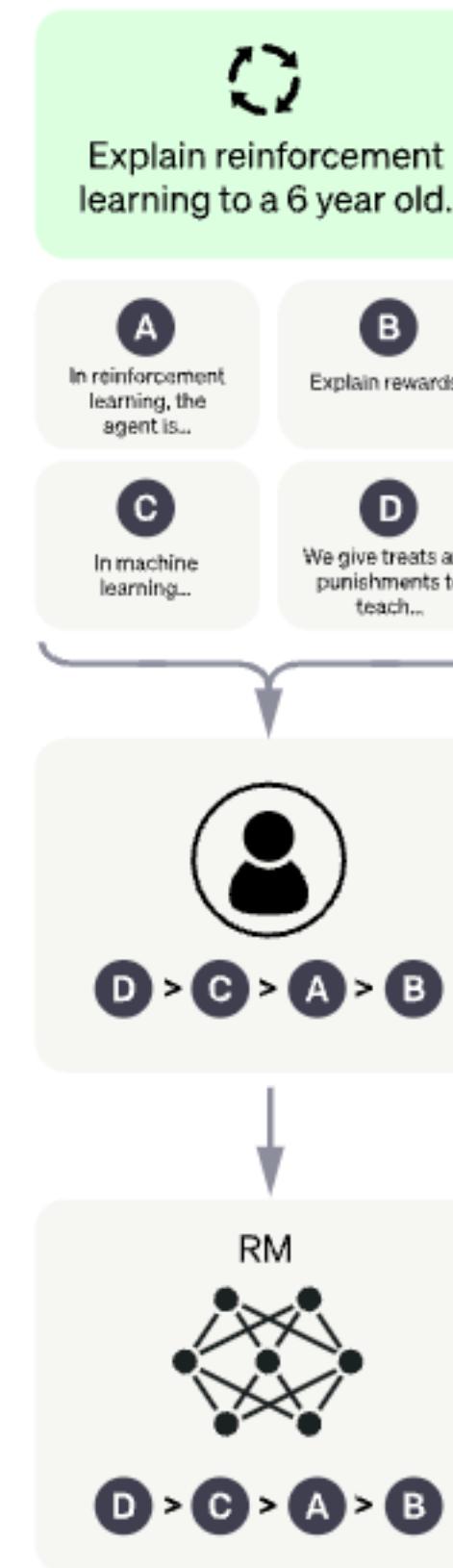
What should we do instead?

Training a reward model

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.

- 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)

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

Training a reward model

Train the reward model to give higher scores to “good” demonstrations

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



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

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

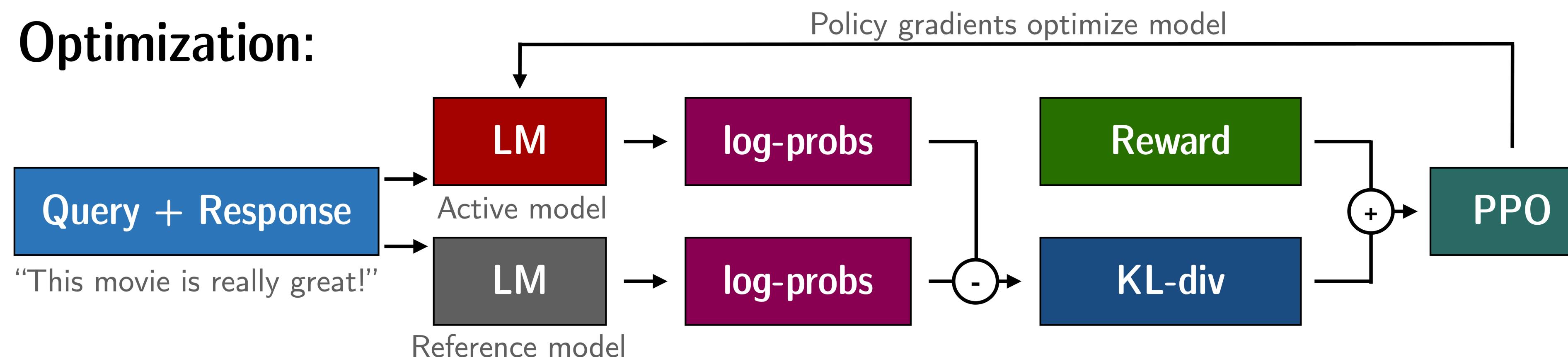
Rollout:



Evaluation:



Optimization:

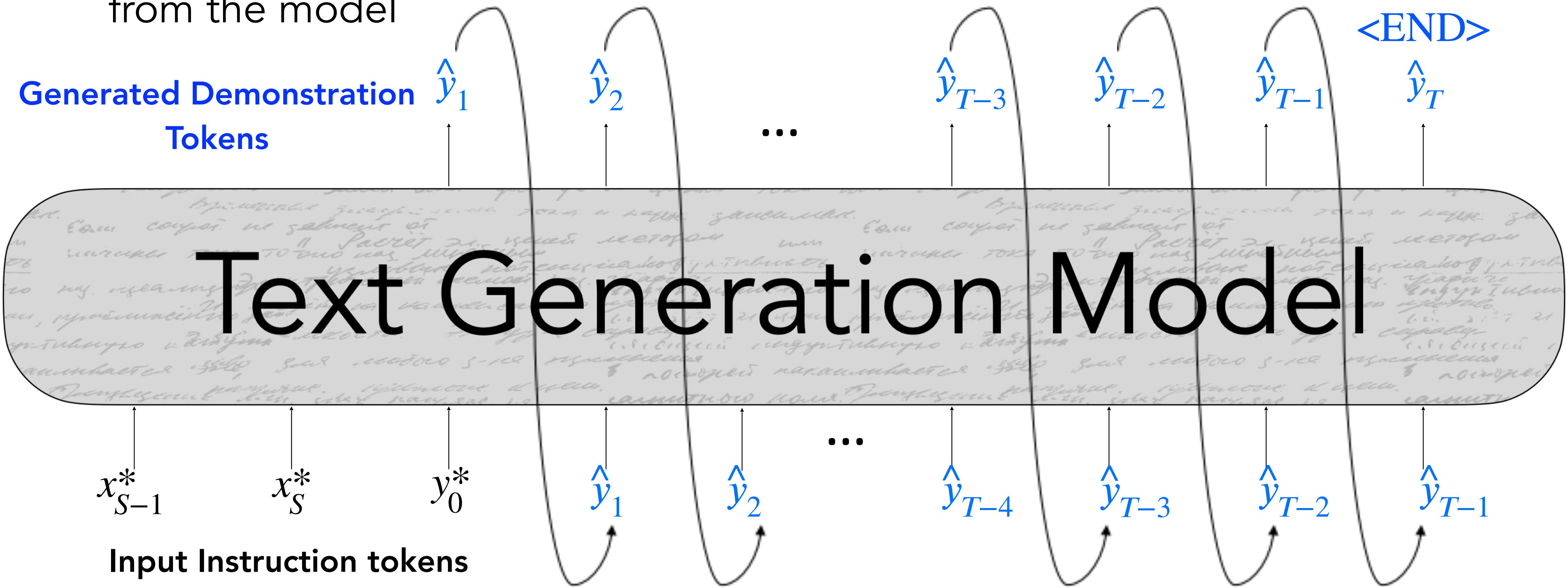


What reinforcement learning
algorithm have we already seen?

REINFORCE

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

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



REINFORCE: Basics

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

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

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

...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

$$\mathcal{L}_{RL} = -(\textcolor{blue}{R}(\hat{Y}) - b) \sum_{t=1}^T \log \textcolor{violet}{P}(\hat{y}_t | \{x^*\}; \{\hat{y}\}_{<t})$$

- Mix with MLE

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

REINFORCE vs. PPO

- REINFORCE

$$\mathcal{L}_{RL} = - (R(\hat{Y}) - b) \sum_{t=1}^T \log \textcolor{violet}{P}(\hat{y}_t | \{x^*\}; \{\hat{y}\}_{<t})$$

- PPO

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

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

How does PPO differ?

- PPO

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

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

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



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

How does PPO differ?

- PPO

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

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

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

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

How does PPO differ?

- PPO

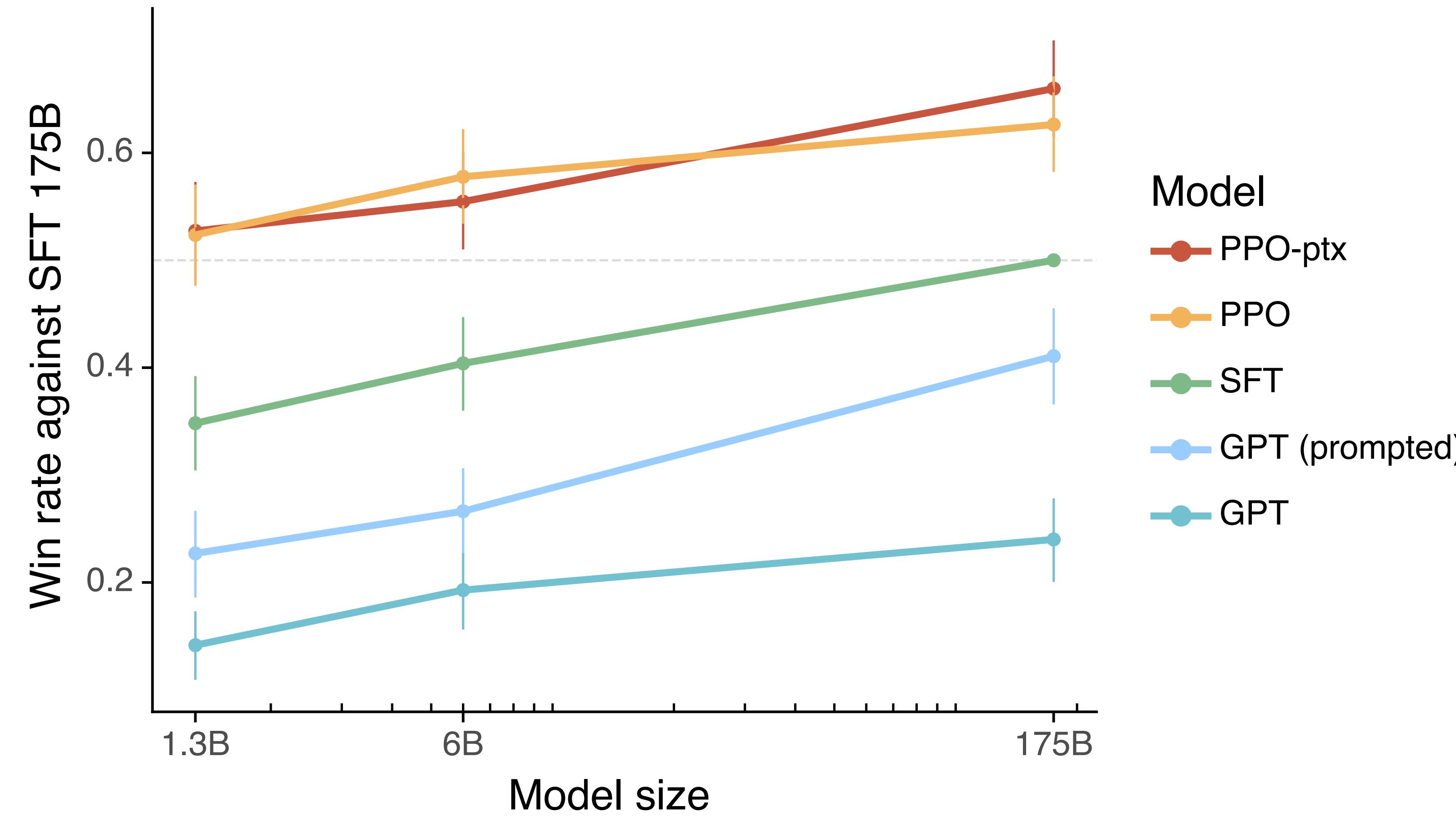
$$\mathcal{L}_{PPO} = - \min \left(R(\hat{Y}) \sum_{t=1}^T \log \frac{P_{curr}(\hat{y}_t | \{x^*\}; \{\hat{y}\}_{<t})}{P_{base}(\hat{y}_t | \{x^*\}; \{\hat{y}\}_{<t})}, g(\epsilon, R(\hat{Y})) \right)$$

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

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

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](#)

```
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



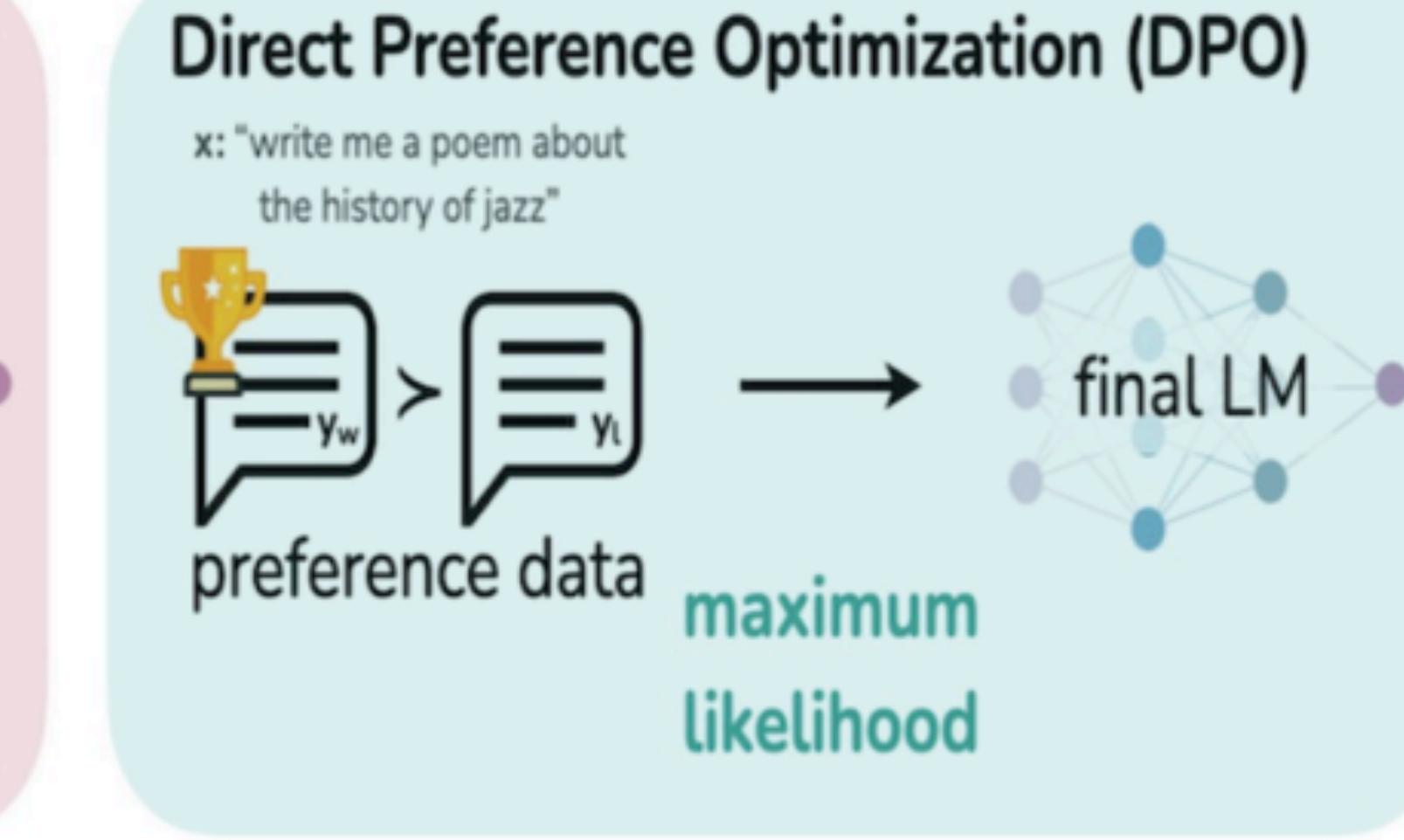
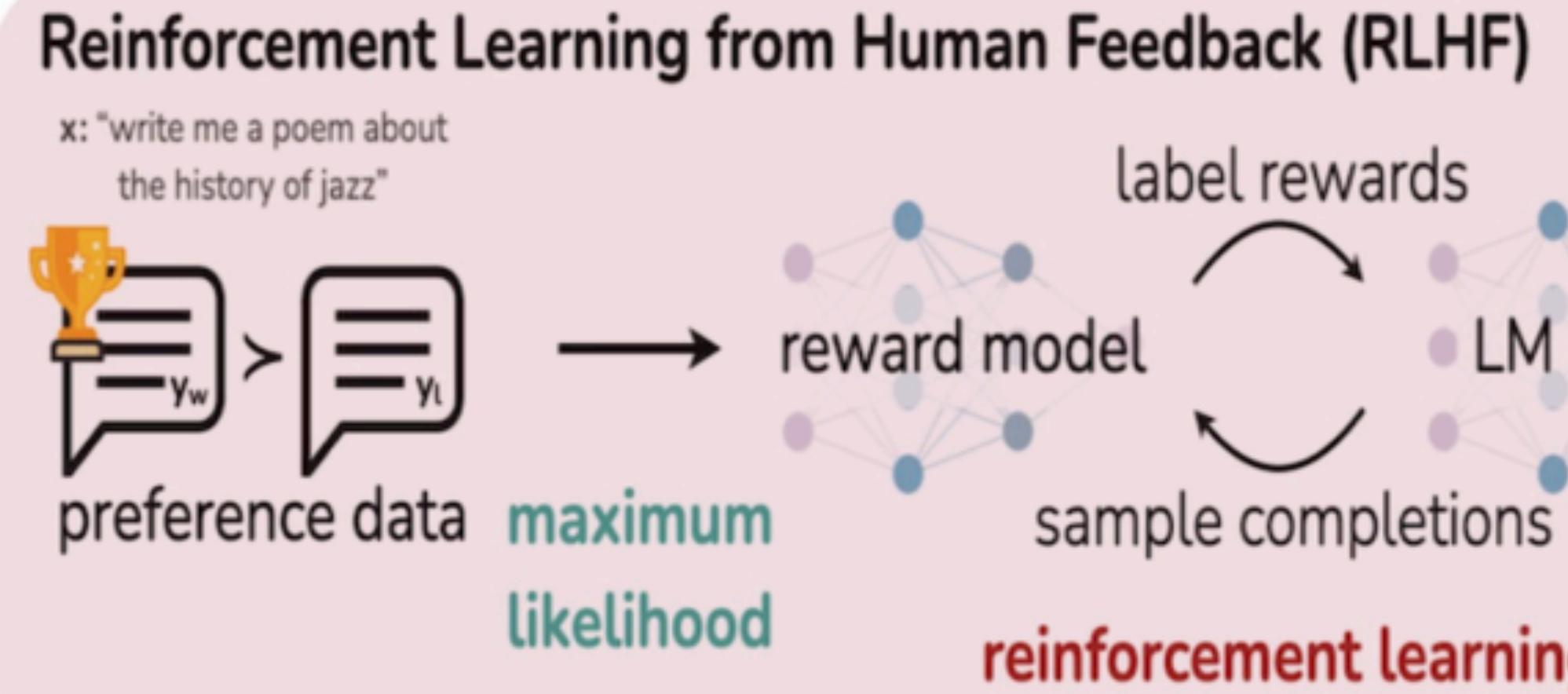
World War 1 was a big war that started in 1914 and lasted until 1918. It was fought between many countries, including Britain, France, and Russia on one side, and Germany, Austria-Hungary, and the Ottoman Empire on the other.

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



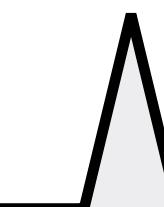
- Reward model in RLHF learns human preferences. In DPO, we optimize for human preferences directly:

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

Direct Preference Optimization

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

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



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

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