CS11-711 Advanced NLP

Fine-tuning

Sean Welleck

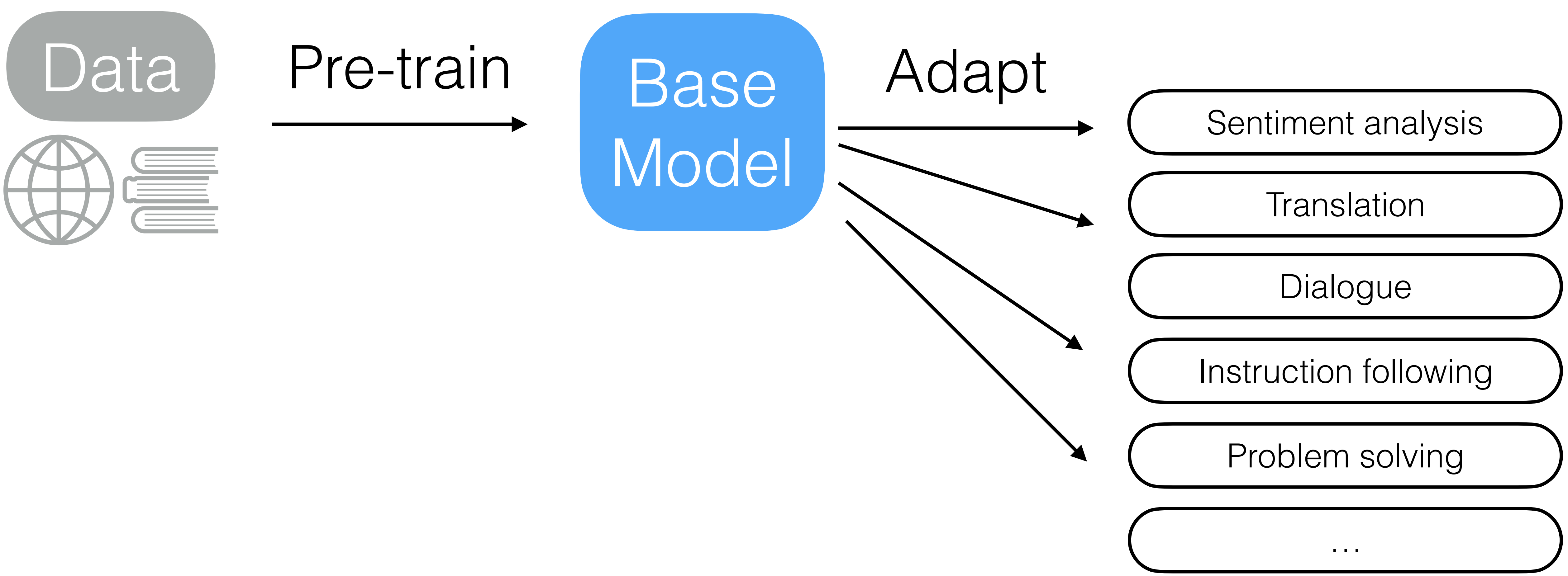
Carnegie Mellon University
Language Technologies Institute

https://cmu-l3.github.io/anlp-spring2025/

https://github.com/cmu-l3/anlp-spring2025-code

Some slides adapted from Graham Neubig (Spring 2024) and Rishabh Agarwal's tutorial

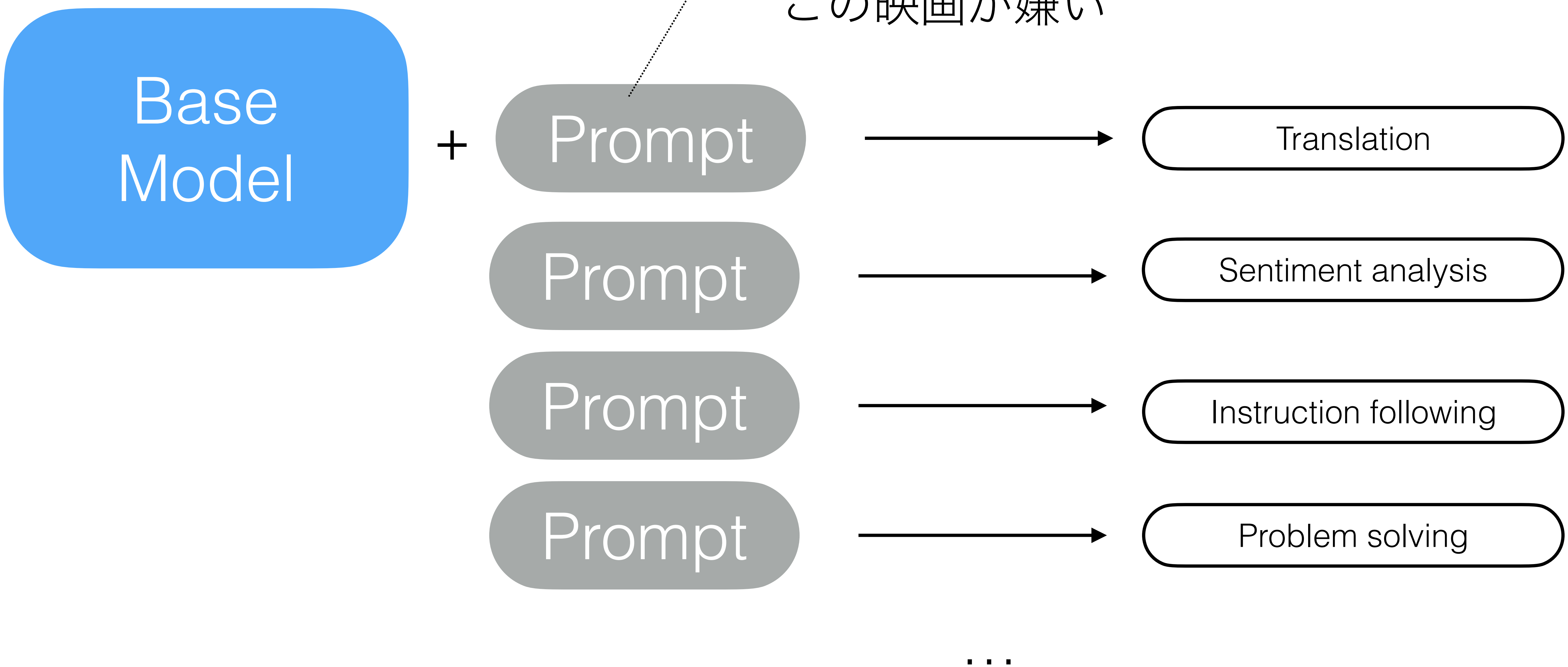
Recap: Pre-training



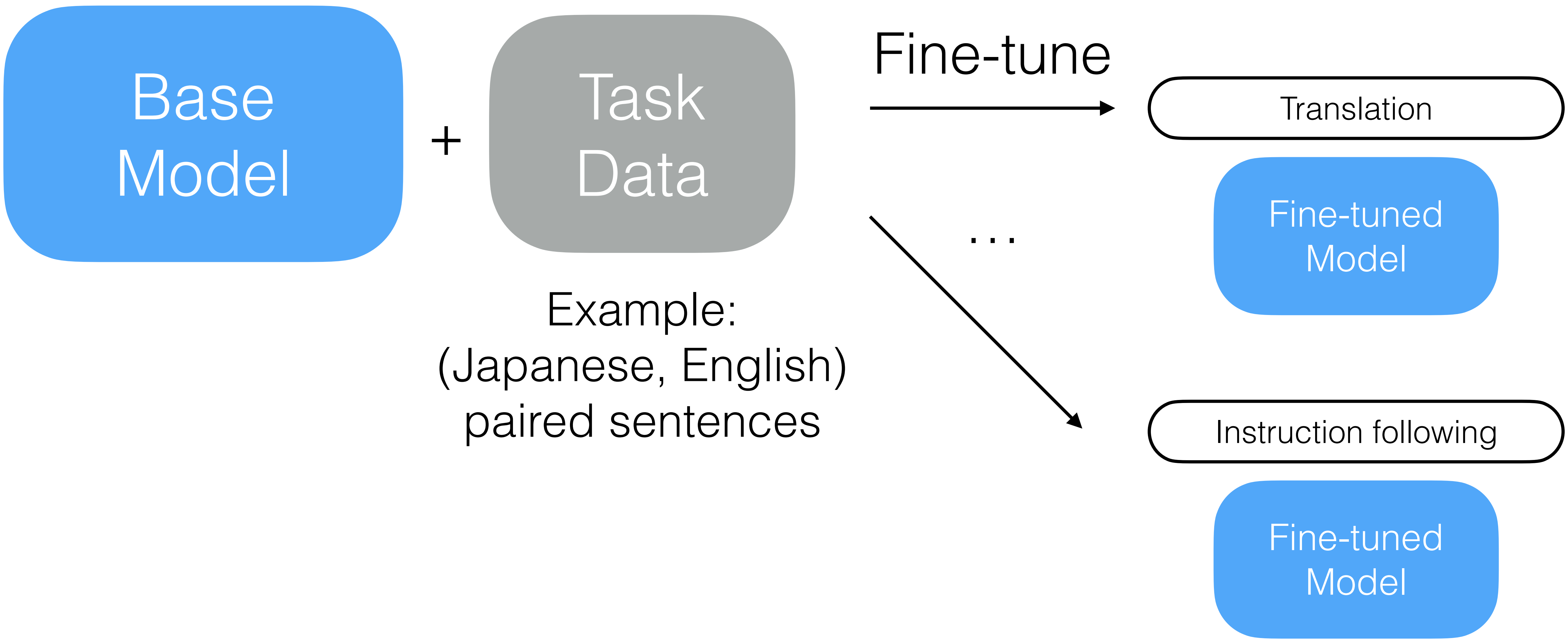
Recap: prompting

Lecture 8

Example:
“Translate this sentence into English:
この映画が嫌い”



Today: fine-tuning



Today's lecture

- Fine-tuning basics
- Instruction tuning
- Knowledge distillation
- Efficient fine-tuning

Today's lecture

- **Fine-tuning basics**
  - Standard language model fine-tuning
  - Effects of fine-tuning

Standard LM fine-tuning

- Initialize a model $p_\theta(y \mid x)$ with a pre-trained model
- Train the model on a dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$

$$\arg\min_{\theta} \sum_{(x,y)\in D} \sum_{t} -\log p_\theta(y_t \mid y_{<t}, x)$$

Use cross-entropy loss
(Lectures 3, 4, 6)

Standard LM fine-tuning

- In summary:

$$p_\theta = \text{Finetune}(\text{pre-trained model}, D, \mathscr{L}_{CE})$$

In general, fine-tuning refers to adjusting a pre-trained model's parameters by optimizing a loss function (not necessarily a language model / $\mathscr{L}_{CE}$)

Examples

X

Y

Article (paper)

Abstract


```
{
    "abstract": "\" we have studied the leptonic decay @xmath0 , vi
    "article": "\"the leptonic decays of a charged pseudoscalar mes
    "section_names": "[sec:introduction]introduction\n[sec:detector]
}
```


Training set: 300,000+ examples

https://huggingface.co/datasets/armanc/scientific_papers

Examples

X

Article (paper)
Problem

Y

Abstract
Solution


```
{'problem': 'A board game spinner is divided into three parts labele
 'level': 'Level 1',
 'type': 'Counting & Probability',
 'solution': 'The spinner is guaranteed to land on exactly one of t
```


Training set: 7,500 examples

https://huggingface.co/datasets/hendrycks/competition_math

Examples

X

Article (paper)
Problem
Prompt

Y

Abstract
Solution
Response

question string · *lengths* 12↔4.07k 96.5%	response string · *lengths* 0↔1.5k 92.9%
How does the sentence end?…	B). Rebecca was saddened by people's vile insults. Here's why: "Thicker skin" is a way to say that…
Write a sentence not in English.	Voici une phrase en français: "Le chat mange la souris." Explication pour un enfant de cinq ans: Cette phrase n'est pas en anglais, mais en français. Elle dit que le chat (l'animal avec des moustaches et qui dit "miaou") mange la souris (un petit animal avec une longue queue).

Training set: 2,910,000 examples

https://huggingface.co/datasets/Open-Orca/OpenOrca

Code example

```
model = "HuggingFaceTB/SmolLM2-135M"

tokenizer = AutoTokenizer.from_pretrained(model)
model = AutoModelForCausalLM.from_pretrained(model)
```

```
for epoch in range(num_epochs):

    model.train()
    total_loss = 0

    for i, batch in enumerate(train_loader):
        optimizer.zero_grad()
        input_ids = batch["input_ids"]
        attention_mask = batch["attention_mask"]

        outputs = model(input_ids, attention_mask=attention_mask, labels=input_ids)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
```

https://github.com/cmu-l3/anlp-spring2025-code/tree/main/09_finetuning

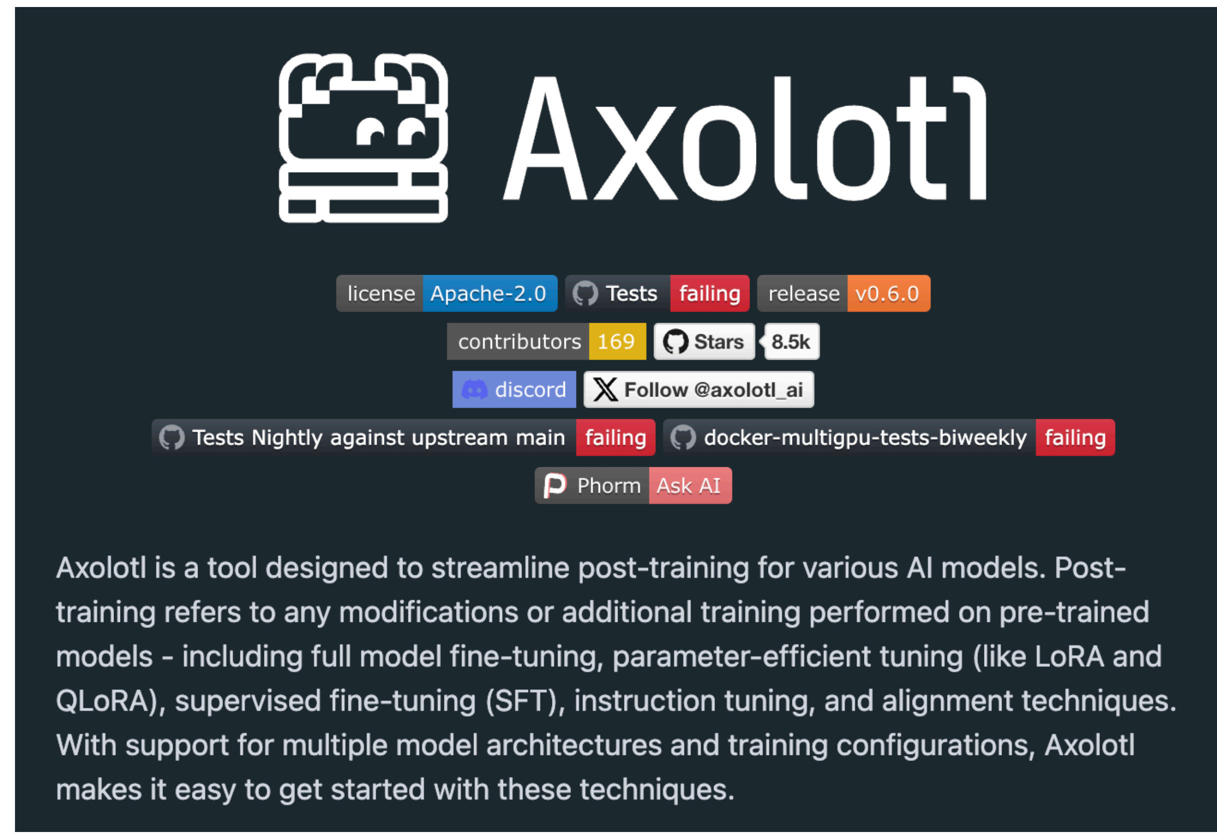
Code example

```python
for epoch in range(starting_epoch, args.num_train_epochs):
    model.train()
    if args.with_tracking:
        total_loss = 0
    if args.resume_from_checkpoint and epoch == starting_epoch and resume_step is not None:
        # We skip the first `n` batches in the dataloader when resuming from a checkpoint
        active_dataloader = accelerator.skip_first_batches(train_dataloader, resume_step)
    else:
        active_dataloader = train_dataloader
    for step, batch in enumerate(active_dataloader):
        with accelerator.accumulate(model):
            outputs = model(**batch)
            loss = outputs.loss
            # We keep track of the loss at each epoch
            if args.with_tracking:
                total_loss += loss.detach().float()
            accelerator.backward(loss)
            optimizer.step()
            lr_scheduler.step()
            optimizer.zero_grad()

        # Checks if the accelerator has performed an optimization step behind the scenes
        if accelerator.sync_gradients:
            progress_bar.update(1)
            completed_steps += 1

        if isinstance(checkpointing_steps, int):
            if completed_steps % checkpointing_steps == 0 and accelerator.sync_gradients:
                output_dir = f"step_{completed_steps}"
                if args.output_dir is not None:
                    output_dir = os.path.join(args.output_dir, output_dir)
                accelerator.save_state(output_dir)
        if completed_steps >= args.max_train_steps:
            break
```

https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_clm_no_trainer.py

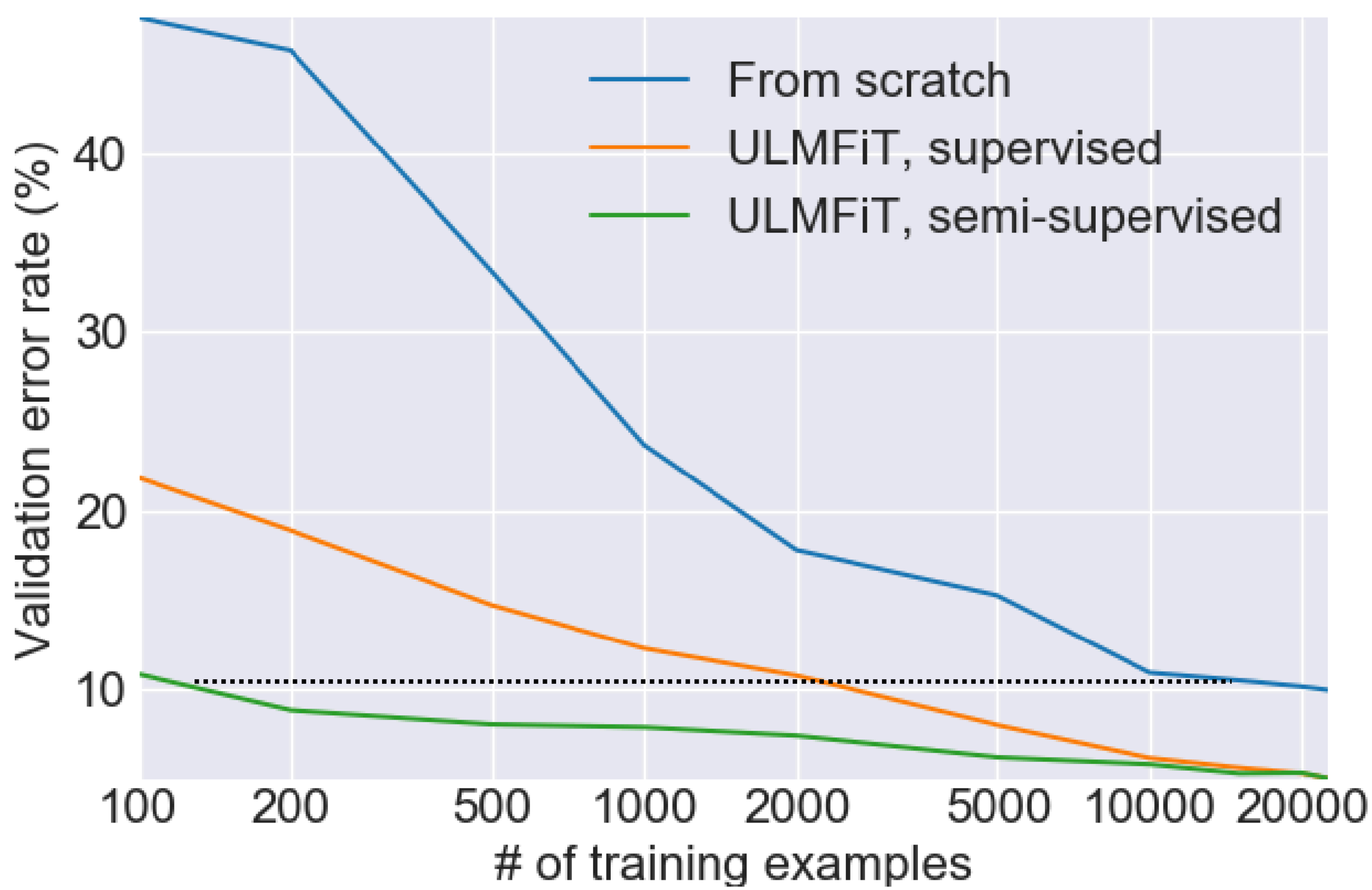
“No code” fine-tuning libraries

- Llama Factory, Axolotl



Effects of fine-tuning

- Starting from a pre-trained model is data-efficient



Howard & Ruder 2018

Effects of fine-tuning

- "Narrows" the distribution
  - Pre-training: minimize $D_{KL}(p_{data}, p_\theta)$
  - Fine-tuning: minimize $D_{KL}(p_{data\ finetune}, p_\theta; p_0)$
- Typically the pretraining data will cover a wider distribution than the fine-tuning data

D_{KL} connection: Lecture 3

Effects of fine-tuning

- Example symptoms:
  - Summarization model doesn't work well on translation
  - Model trained with specific formatting requires the formatting
  - Model can't few-shot learn well after fine-tuning

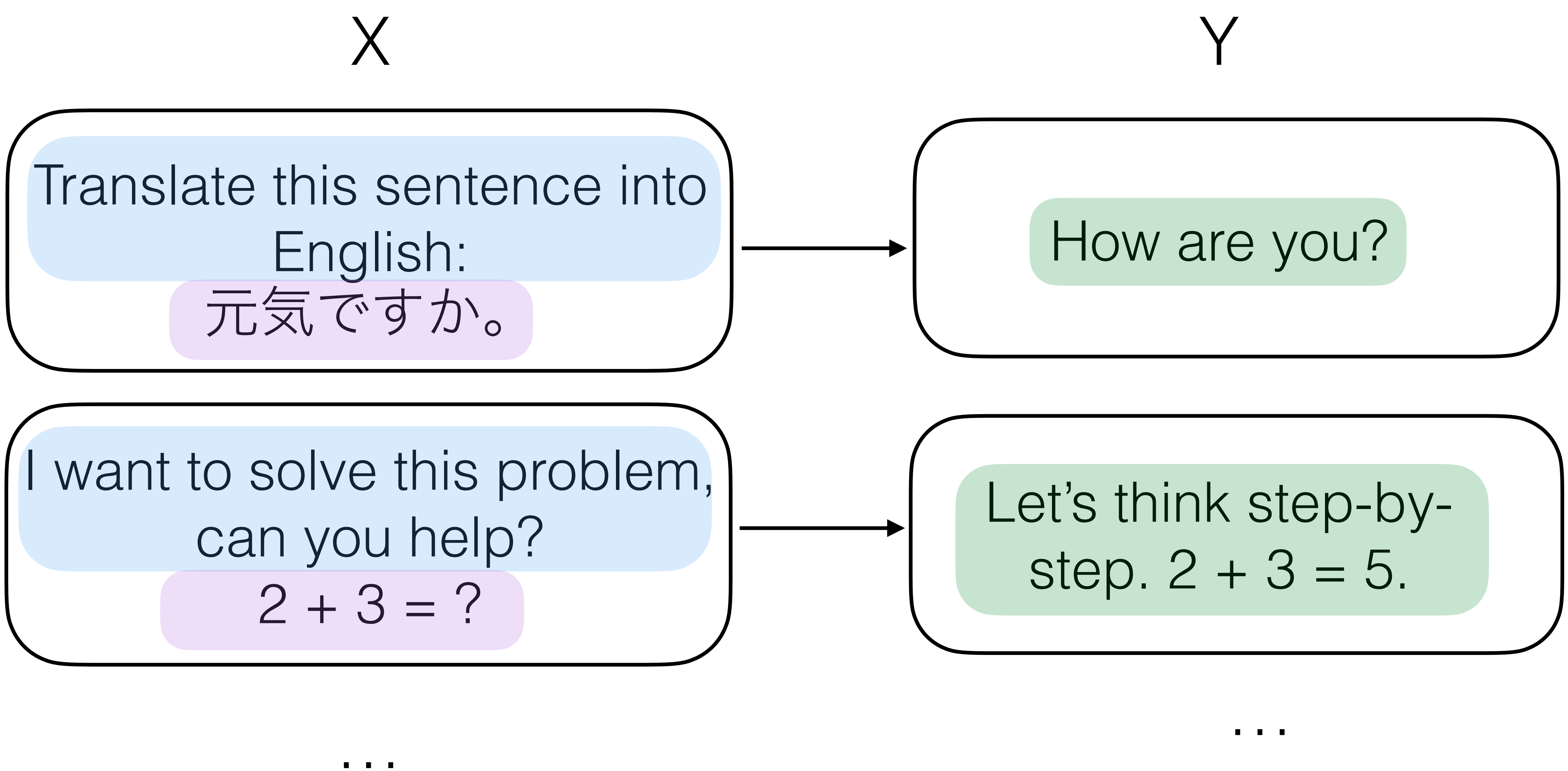
  …

Today's lecture

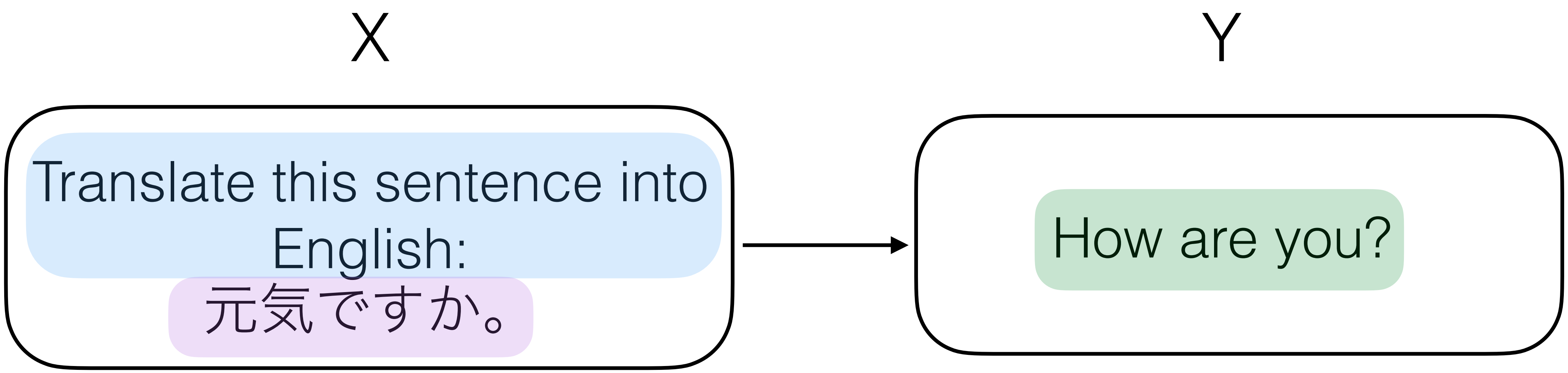
- Fine-tuning basics
- **Instruction tuning**
  - Chat tuning

Basic idea

- Fine-tune a model to perform multiple tasks
- Insight: use (instruction + input, output) data



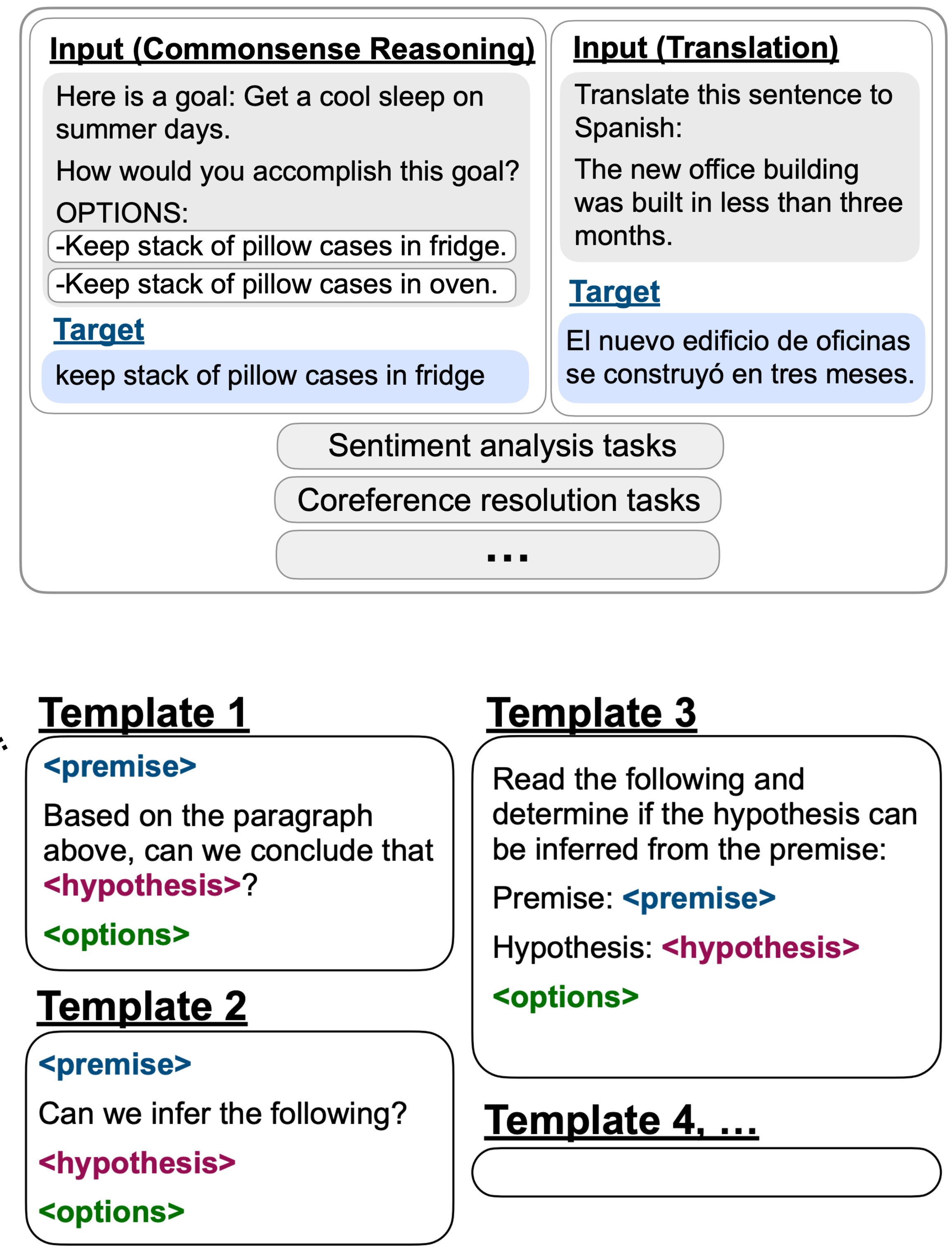
Variations



- Instructions: template, human, model-generated
- Input: dataset, human, model-generated
- Output: dataset, human, model-generated
- Domain: general, code, math, chat, …

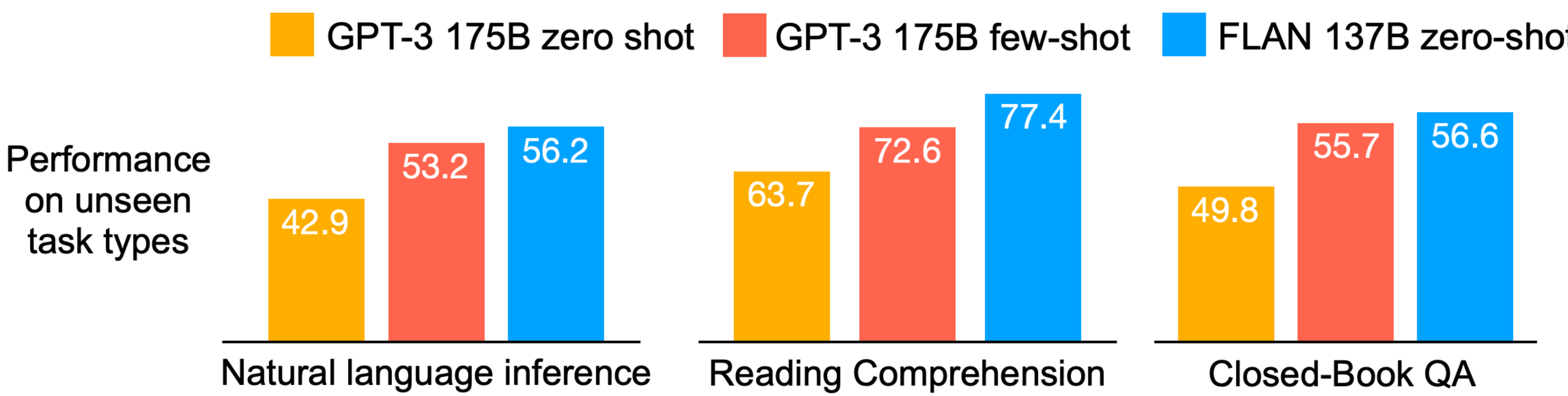
Example: FLAN [Wei et al 2021]

- 62 NLP datasets
- Instructions: templates
- Input: from dataset
- Output: from dataset



Example: FLAN [Wei et al 2021]

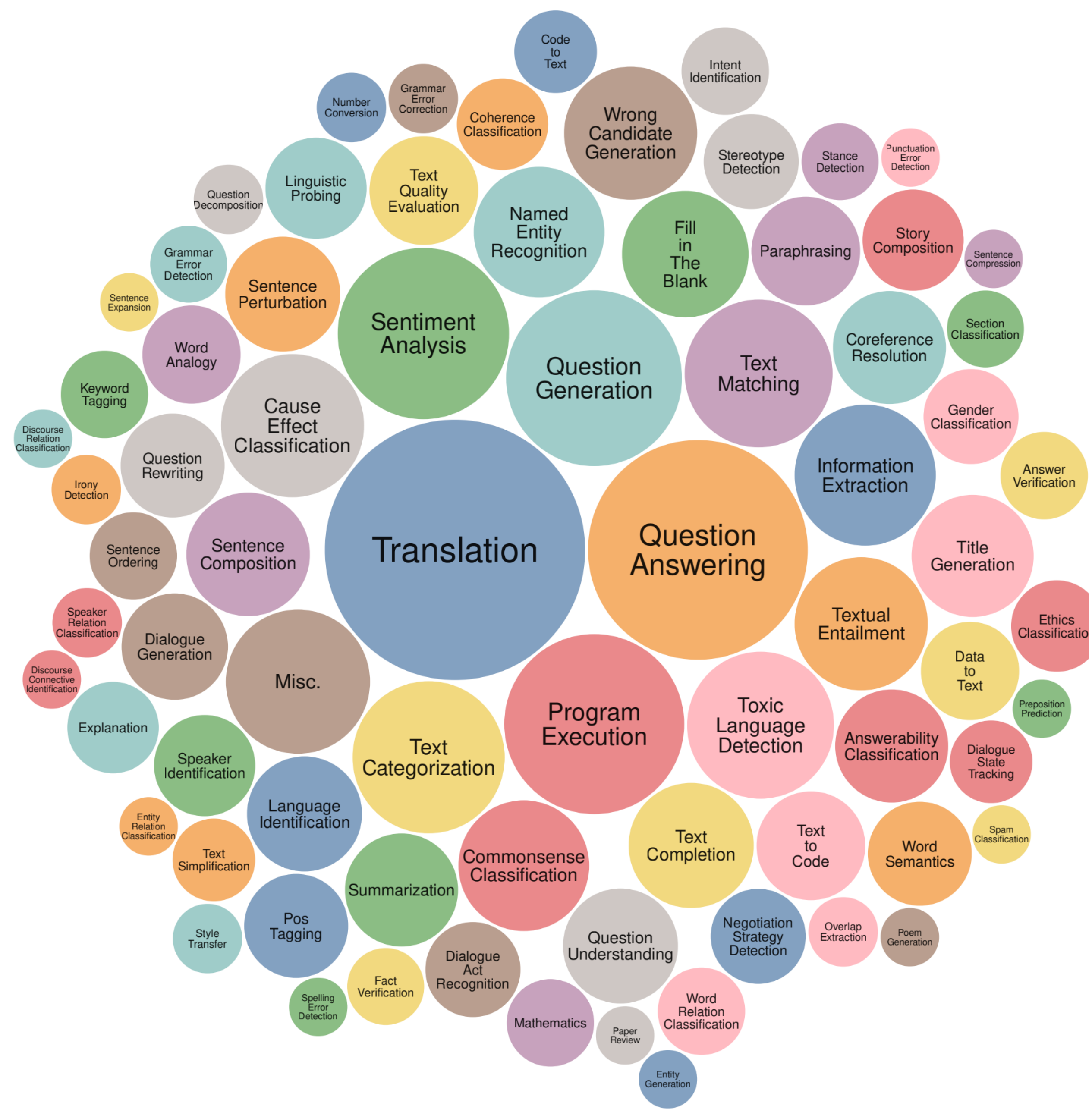
- Key finding: model can generalize to unseen tasks



Example: SuperNaturalInstructions

[Mishra et al 2021, Wang & Mishra et al 2022]

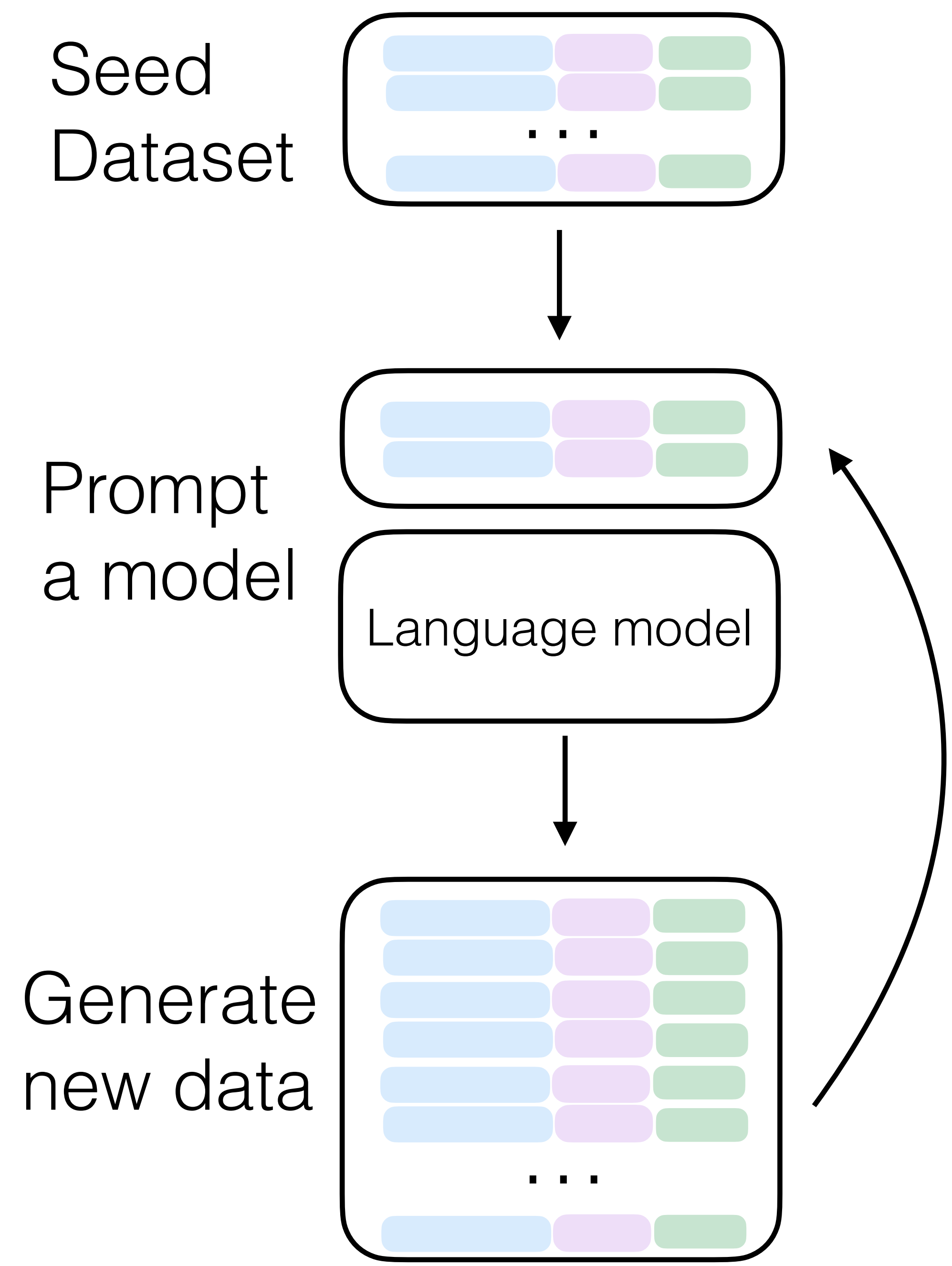
- 1,600 tasks
- Instructions: crowdsourced
- Input: crowdsourced
- Output: crowdsourced



Example: Self-Instruct

[Mishra et al 2021, Wang & Mishra et al 2022]

- 50,000+ instructions
- Instructions: model
- Input: model
- Output: model



Example: Self-Instruct

[Mishra et al 2021, Wang & Mishra et al 2022]

- 50,000+ instructions
- Instructions: model
- Input: model
- Output: model

Instruction: Given an address and city, come up with the zip code.
Input:
Address: 123 Main Street, City: San Francisco
Output: 94105

Instruction: I am looking for a job and I need to fill out an application form. Can you please help me complete it?
Input:
Application Form:
Name:_________ Age:__________ Sex:________
Phone Number:______ Email Address:________
Education:___________________ ...
Output:
Name: John Doe Age: 25 Sex: Male
Phone Number: ...

Instruction: How to write a code for converting degrees fahrenheit to celsius.
Input: Null
Output:

```
def convert_fahrenheit_to_celsius(fahr):
    celsius = (fahr - 32) * 5 / 9
    return celsius
```

Recap: instruction tuning

- Finetune on (instruction, input, output) examples
  - Get data from humans or model generations
- Resulting model can perform many tasks
- Lots of followup work on instruction-tuning data for certain domains: general, code, math, **chat**, …

Chat tuning

- Ultimately, format a chat as a sequence of tokens
  - System prompt
  - [user, assistant, user, assistant, …]
- “Instruction+input” are implicitly in the conversation

System

User

Assistant

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful AI assistant for travel tips and
recommendations<|eot_id|>
<|start_header_id|>user<|end_header_id|>

What can you help me with?<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>
```

Lecture 8

Chat tuning | system prompts

- Example: OpenOrca

System Instruction: You are an AI assistant. User will you give you a task. Your goal is to complete the task as faithfully as you can. While performing the task think step-by-step and justify your steps.

User Instruction: Use the given data to calculate the median.

Input: [7, 3, 8, 2, 10]

Output: To calculate the median, I will follow these steps:
Step 1: Arrange the data in ascending order [2, 3, 7, 8, 10]
Step 2: Determine if the data set has an odd or even number of values
The data set has 5 values, which is an odd number.
Step 3: Find the middle value

Since the data set has an odd number of values, the median is the middle value in the ordered list. The middle value in the list [2, 3, 7, 8, 10] is 7.

System Instruction: You are an AI assistant. Provide a detailed answer so user don't need to search outside to understand the answer.

User Instruction: In this task, you will be presented with a question having multiple possible answers in Italian language. And you should choose a most suitable option out of "A", "B", "C", "D", and "E" based on your commonsense knowledge.

Input: Solve this question: Dove non riusciresti a vedere la luce?
Options: A scrivania B frigorifero C sole D universo E attico

Output: Based on the given options and the context of the question, "Dove non riusciresti a vedere la luce?" (Where would you not be able to see light?), the most suitable answer would be: B frigorifero (refrigerator).

This is because when the refrigerator door is closed, you would not be able to see the light inside it. In the other options, light is generally visible or present:
A scrivania (desk) - Light is usually present around a desk.
C sole (sun) - The sun is the primary source of light in our solar system.
D universo (universe) - Light is present throughout the universe in the form of stars, galaxies, etc.
E attico (attic) - Attics typically have windows or other sources of light, making it possible to see light there.

System messages: 16 hand-written

Outputs: from GPT-4

https://huggingface.co/datasets/Open-Orca/OpenOrca

Chat tuning | system prompts

- Example: Llama 3
  - Human-written

> You are a helpful and cheerful AI Chatbot that acts as a meal plan assistant for busy families. The family consists of 2 adults, 3 teenagers, and 2 preschoolers. Plan two or three days at a time and use leftovers or extra ingredients for the second day's plan. The user will let you know if they want two or three days. If they don't, assume three days. Each plan should include breakfast, lunch, snack, and dinner. Ask the user if they approve of the plan or need adjustments. After they approve provide a grocery list with family size in mind. Always keep family preferences in mind and if there's something that they don't like provide a substitution. If the user is not feeling inspired then ask them what's the one place they wish they could visit on vacation this week and then suggest meals based on that location's culture. Weekend meals can be more complex. Weekday meals should be quick and easy. For breakfast and lunch, easy food like cereal, English muffins with pre-cooked bacon, and other quick easy foods are preferred. The family is busy. Be sure to ask if they have essentials and favorites on hand like coffee or energy drinks so they don't forget to buy it. Remember to be budget-conscious unless it's a special occasion.

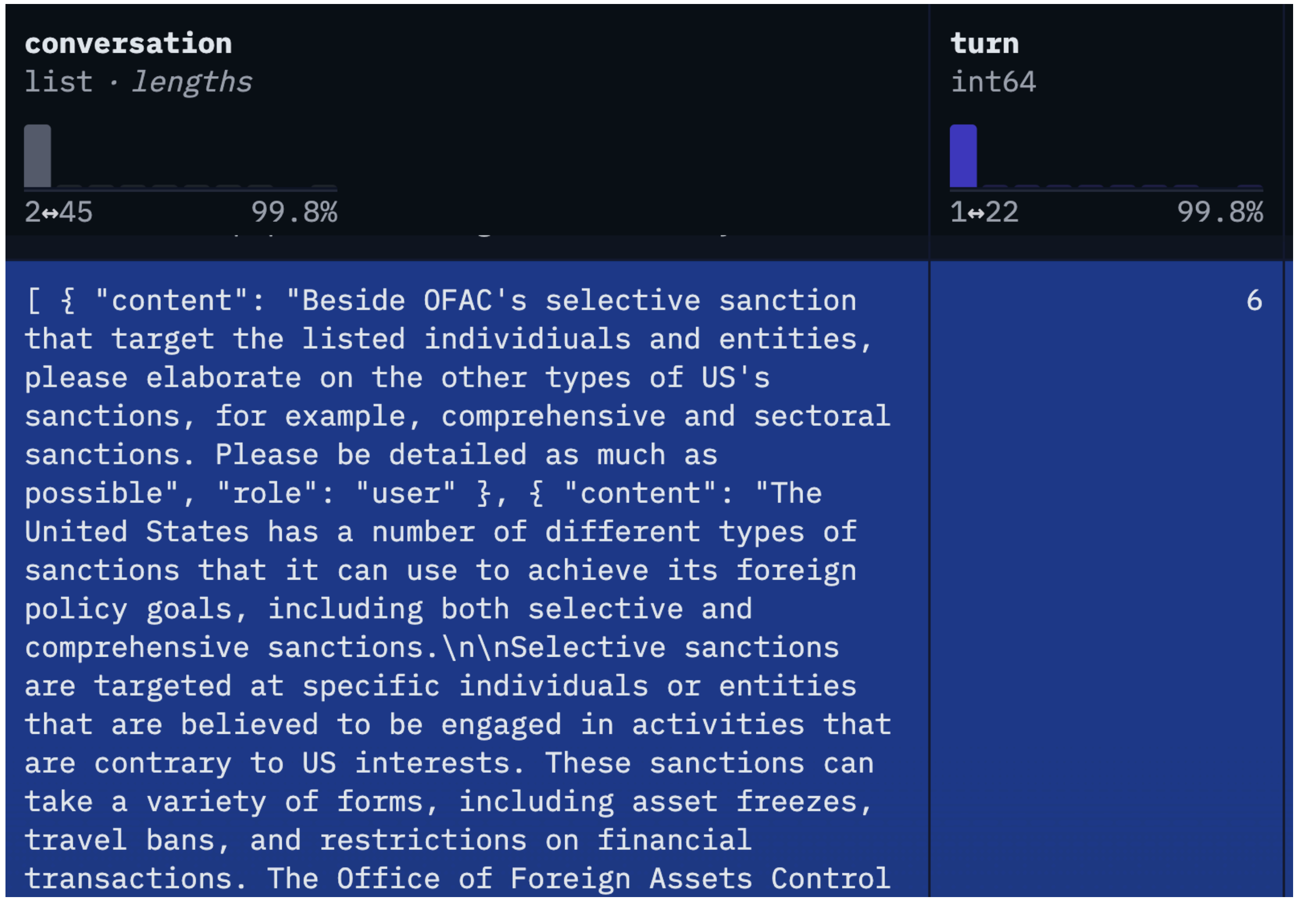
  - Metadata

> **System Prompt**
>
> Environment: **ipython,**
> Tools: **brave_search, wolfram_alpha**
> Cutting Knowledge Date: : **December 2023**
> Today's Date: **Jul 11, 2024**

  - Other mentions of system prompts, no details

Chat tuning | multi-turn

- Example: LMSys-1M



Data source: online LLM service hosted by Berkeley/Stanford

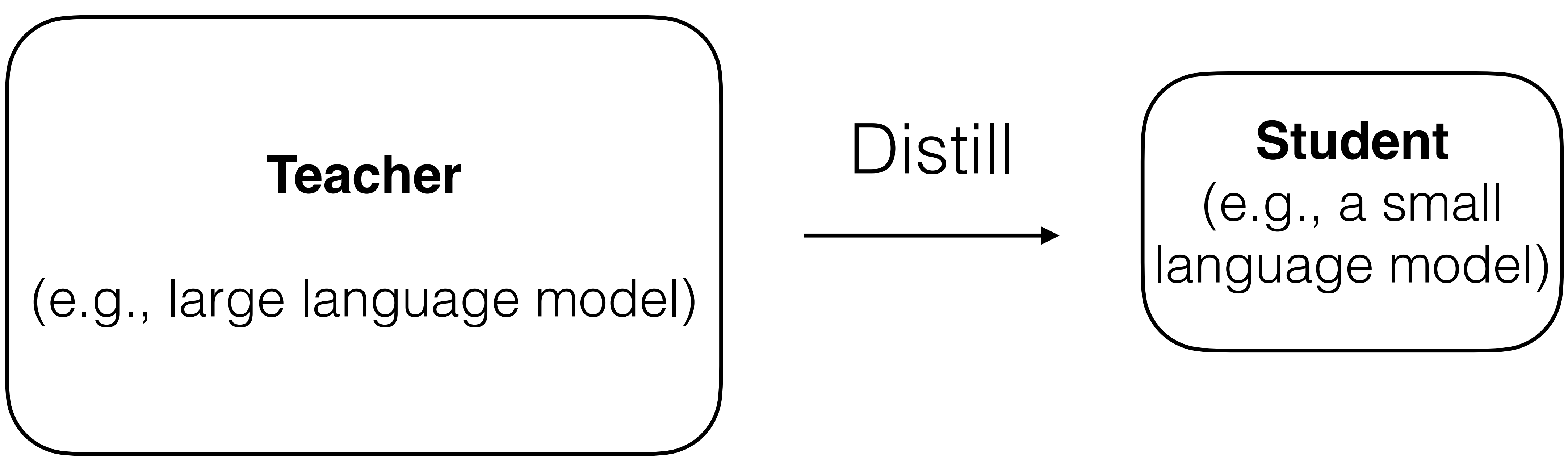
https://huggingface.co/datasets/lmsys/lmsys-chat-1m

Today's lecture

- Fine-tuning basics
- Instruction tuning
- **Knowledge distillation**

Knowledge distillation

- Several methods we discussed use a good model (e.g., GPT-4) to generate data for another model
- Instance of *knowledge distillation* [Hinton et al 2015]



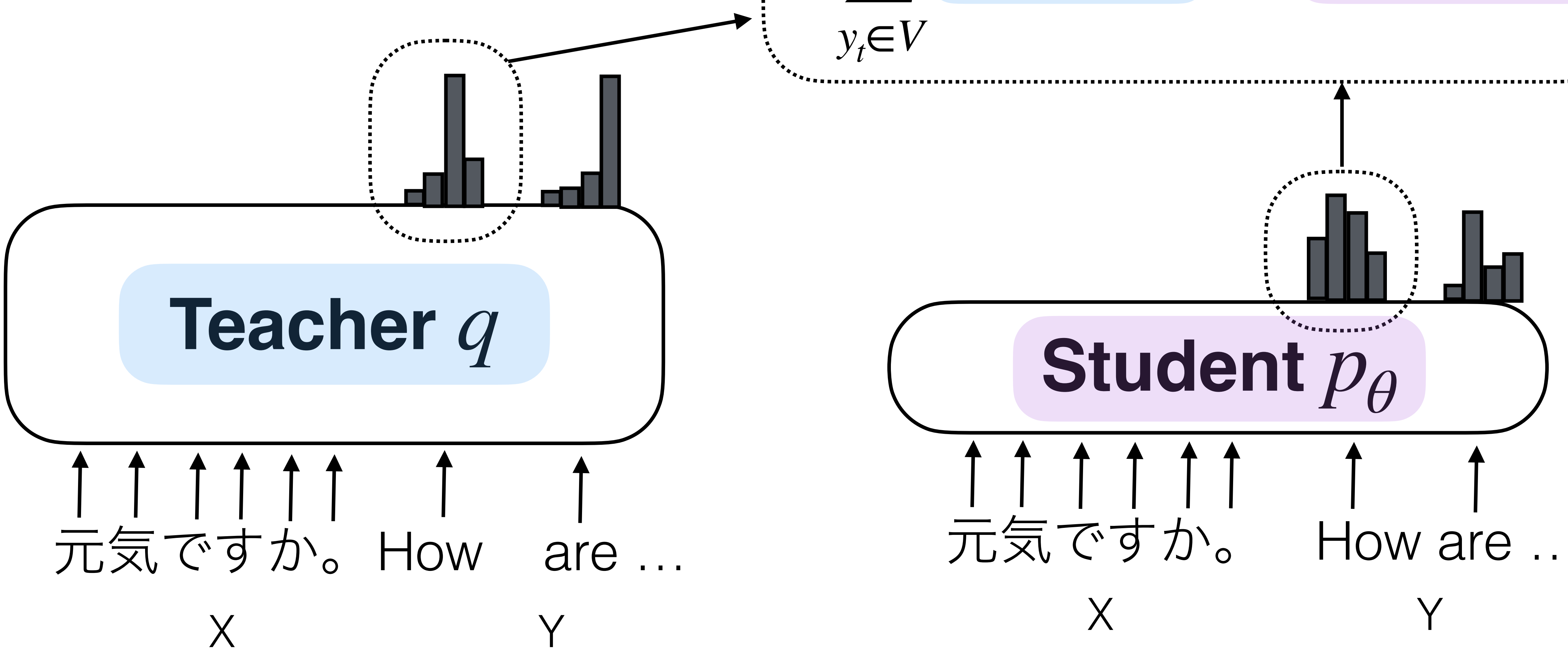
Token-level knowledge distillation

[Hinton et al 2015]

- Train student to mimic teacher's token distributions

Distillation loss (cross entropy)

$$-\sum_{y_t \in V} q(y_t \mid y_{<t}, x) \log p_\theta(y_t \mid y_{<t}, x)$$



Token-level knowledge distillation

[Hinton et al 2015]

- Minimizes KL between teacher and student:

$$\min_{\theta} KL\left(q(y \mid x) \| p_{\theta}(y \mid x)\right)$$

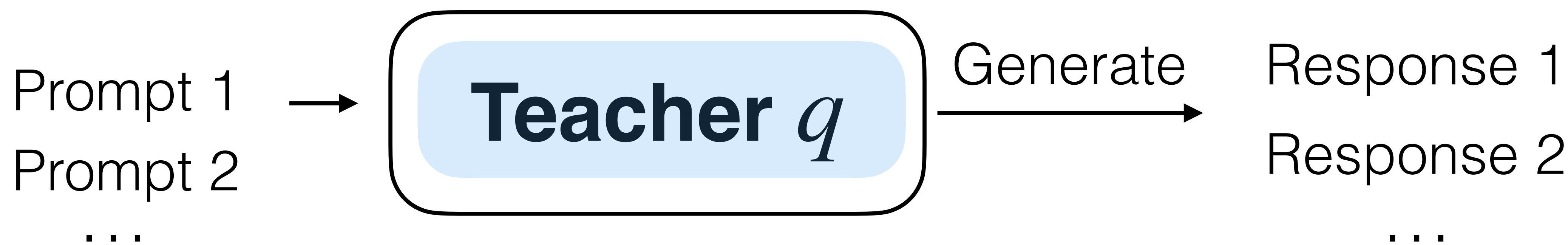
$$\equiv \min_{\theta} \mathbb{E}_{y \sim q(y|x)} \left[\sum_{t} \sum_{y_t \in V} - q(y_t \mid y_{<t}, x) \log p_{\theta}(y_t \mid y_{<t}, x) \right]$$

"Soft labels"

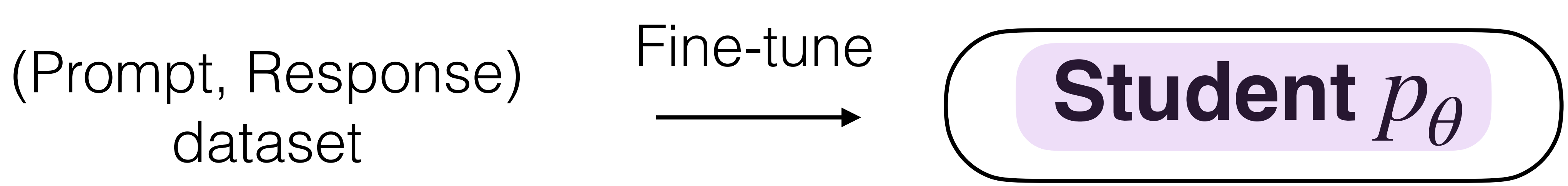
Sequence-level knowledge distillation

[Kim & Rush 2016]

- Generate with a teacher model



- Student model fine-tunes on the generated data



Example: DeepSeek-R1-Distill-Qwen-7B

Sequence-level knowledge distillation

[Kim & Rush 2016]

- Also minimizes KL between teacher and student:

$$\min_{\theta} KL\left(q(y \mid x) \| p_{\theta}(y \mid x)\right)$$

$$\equiv \min_{\theta} \mathbb{E}_{y \sim q(y|x)}\left[-\log p_{\theta}(y \mid x)\right]$$

Teacher generations

Sequence-level knowledge distillation

- [West et al 2022]: the teacher can be an "augmented" language model, e.g.

$$q \propto p_{LLM}(y \mid x) \cdot A(x, y)$$

E.g. a classifier, verifier

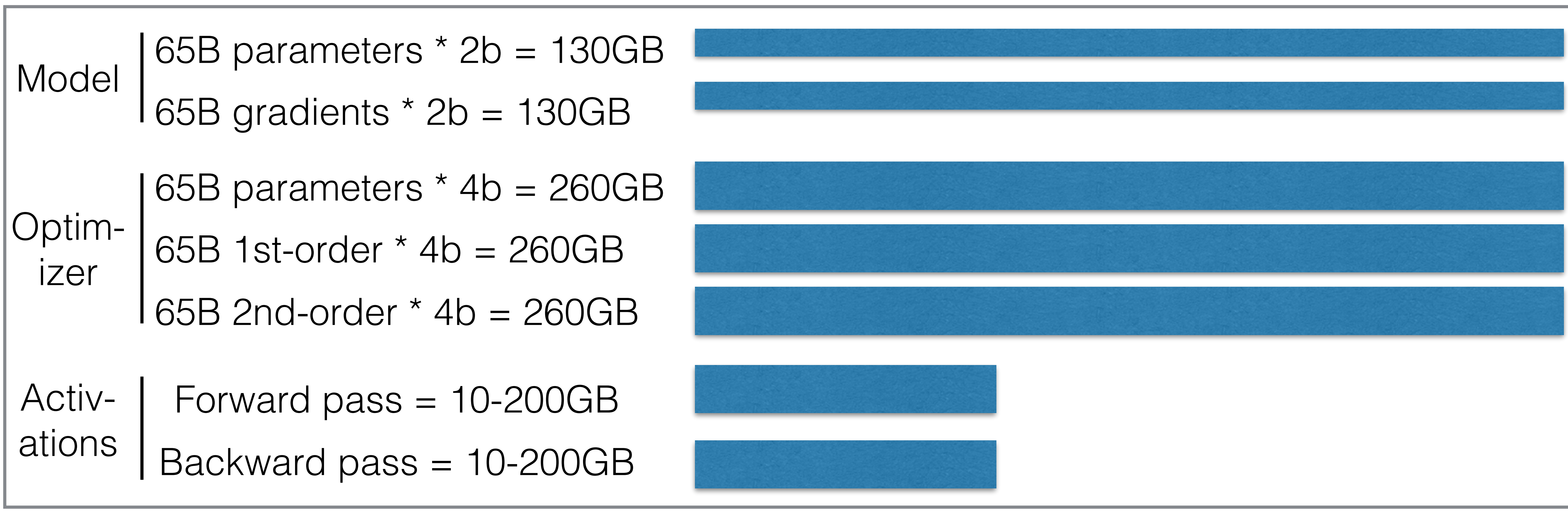
- In principle, if the augmented teacher is better than p_{LLM}, then the student can become better than p_{LLM} through distillation

Today's lecture

- Fine-tuning basics
- Instruction tuning
- Knowledge distillation
- **Efficient fine-tuning**
  - Full fine-tuning
  - Parameter-efficient fine-tuning (LoRA)

Full Fine-tuning

- Simply continue training the LM on the output
- **Issue:** depending on optimizer, optimization method, can take lots of memory!
- **Example:** Training 65B parameter model with 16-bit mixed precision (Rajbhandari et al. 2019)



1000-1400GB of GPU memory!

(can be reduced by using bfloat16, other optimizations)

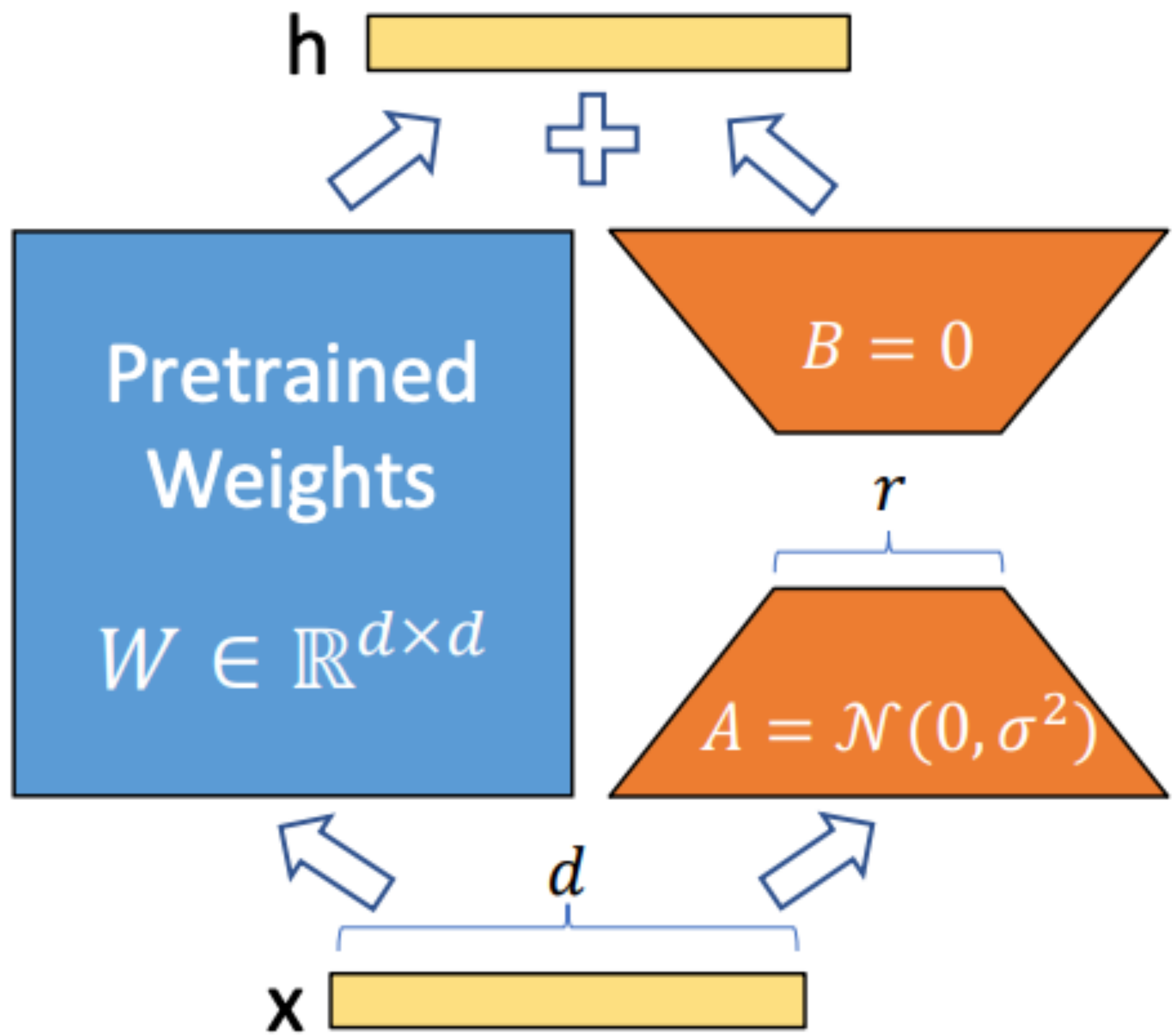
An Aside: GPU Specs

GPU	Memory	Cost (2/2024)	(Cloud) Machines
T40 / K80	24GB	$150	Google Colab, AWS p2.*
V100	32GB	$2,500	Google Colab
A100	40GB or 80GB	$8,000/$16,000	Google Colab, AWS p3.*
H100	80GB	$44,000	AWS p4.*
6000 Ada, L40	48GB	$8000	N/A
Mac M*	Same as CPU	$2000	N/A

Low-Rank Adaptation (LoRA)

[Hu et al 2021]

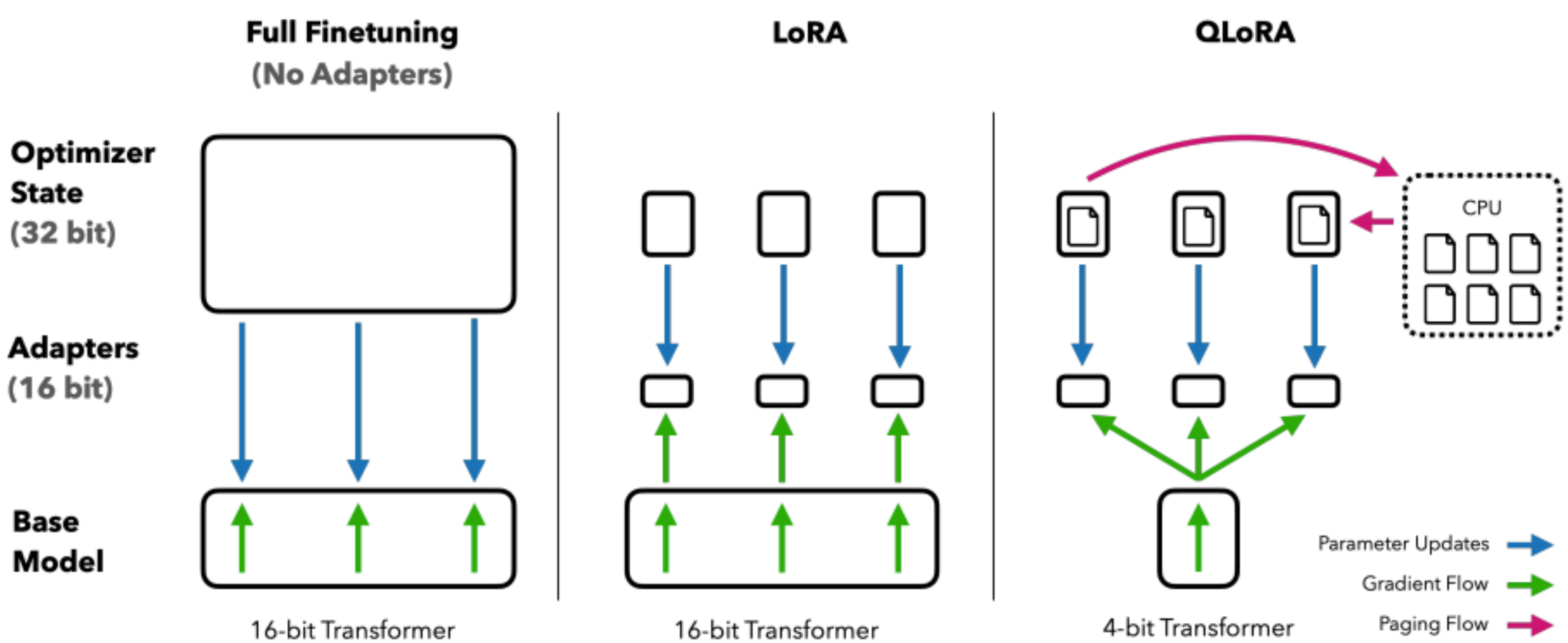
- Freeze pre-trained weights, train low-rank approximation of difference from pre-trained weights
- $$W' = \underbrace{W}_{\mathbb{R}^{d\times d}} + \underbrace{A}_{\mathbb{R}^{d\times r}} \; \underbrace{B}_{\mathbb{R}^{r\times d}}$$
- After training, just add in to pre-trained weights!



Q-LORA

[Dettmers et al 2023]

- Further compress memory requirements for training by
  - 4-bit quantization of the model (later class for details)
  - Use of GPU memory paging to prevent OOM



- Can train a 65B model on a 48GB GPU!

Recap

- Fine-tuning basics
  - Adjust a model's parameters using data
- Instruction tuning
  - Format data so that a model learns to do multiple tasks
- Knowledge distillation
  - Data can come from various teachers (human, model)
- Efficient fine-tuning
  - Only update some of the parameters

Questions?