CS11-711 Advanced NLP

Fine-Tuning

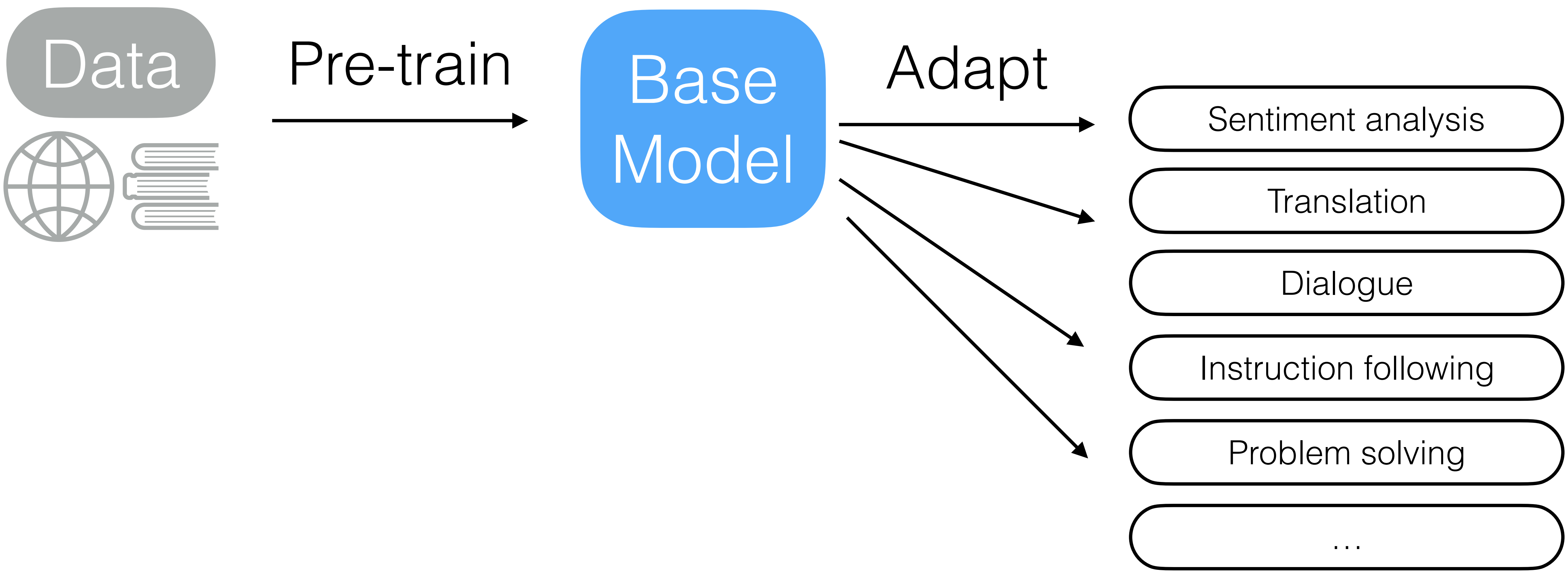
Sean Welleck

https://cmu-l3.github.io/anlp-spring2026/

https://github.com/cmu-l3/anlp-spring2026-code

Recap: Pre-training

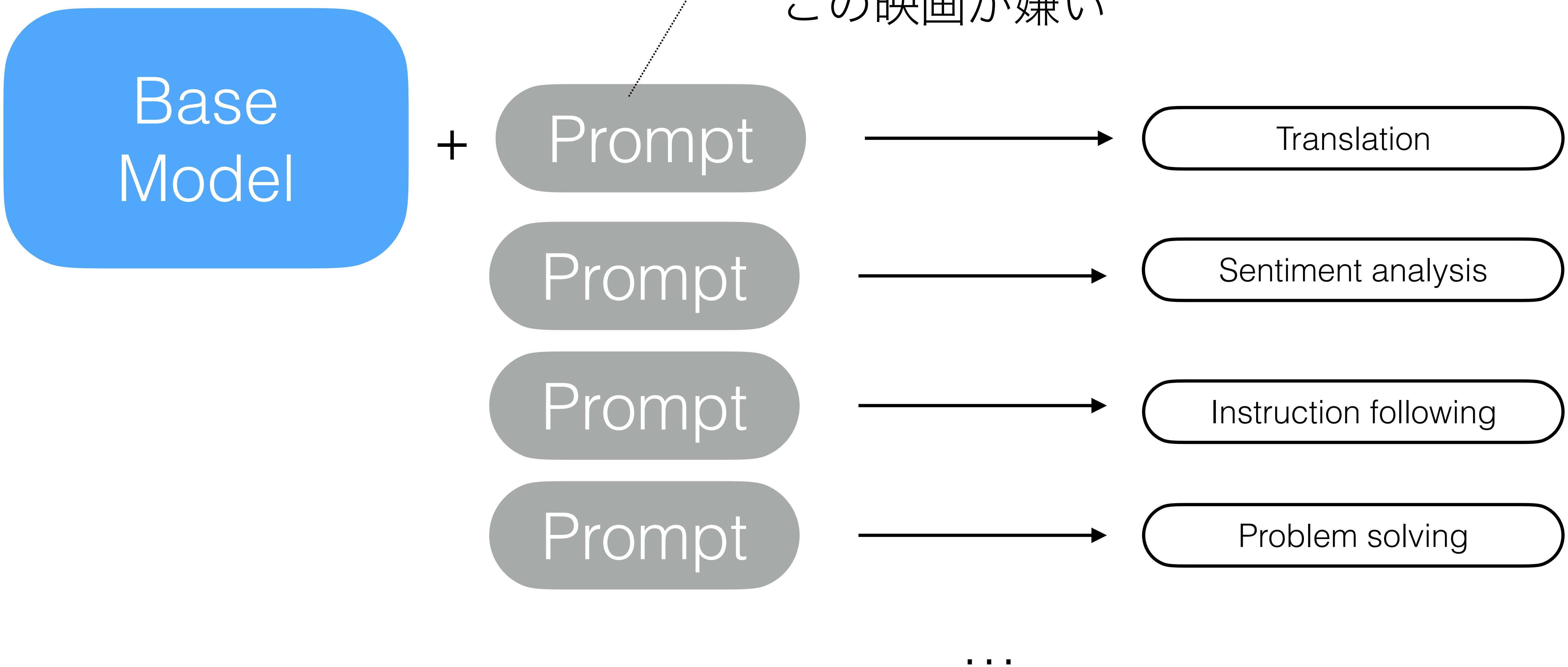
Lecture 6



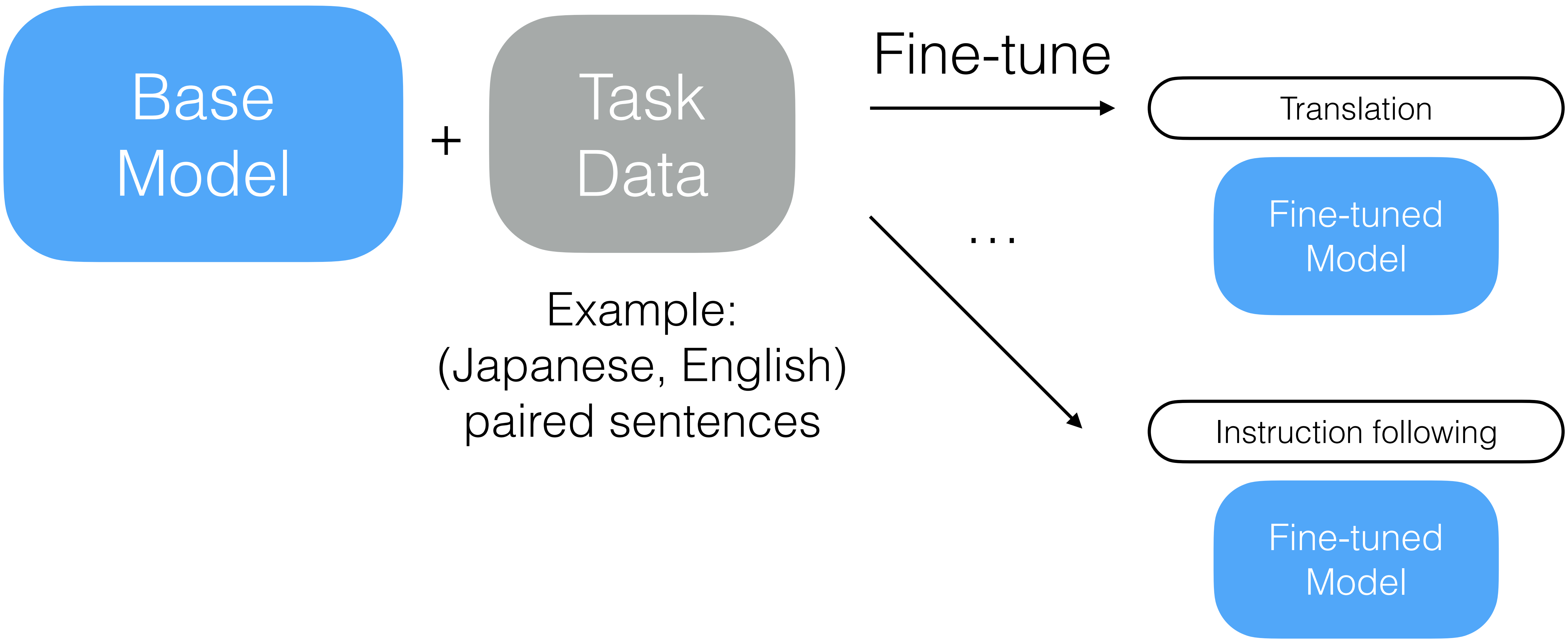
Recap: prompting

Lecture 7

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

Fine-tuning

- **Fine-tuning**: continued gradient-based training of a pre-trained model

- Given pre-trained parameters θ_0 and data $D = \{(x, y)_n\}_{n=1}^{N}$, solve:

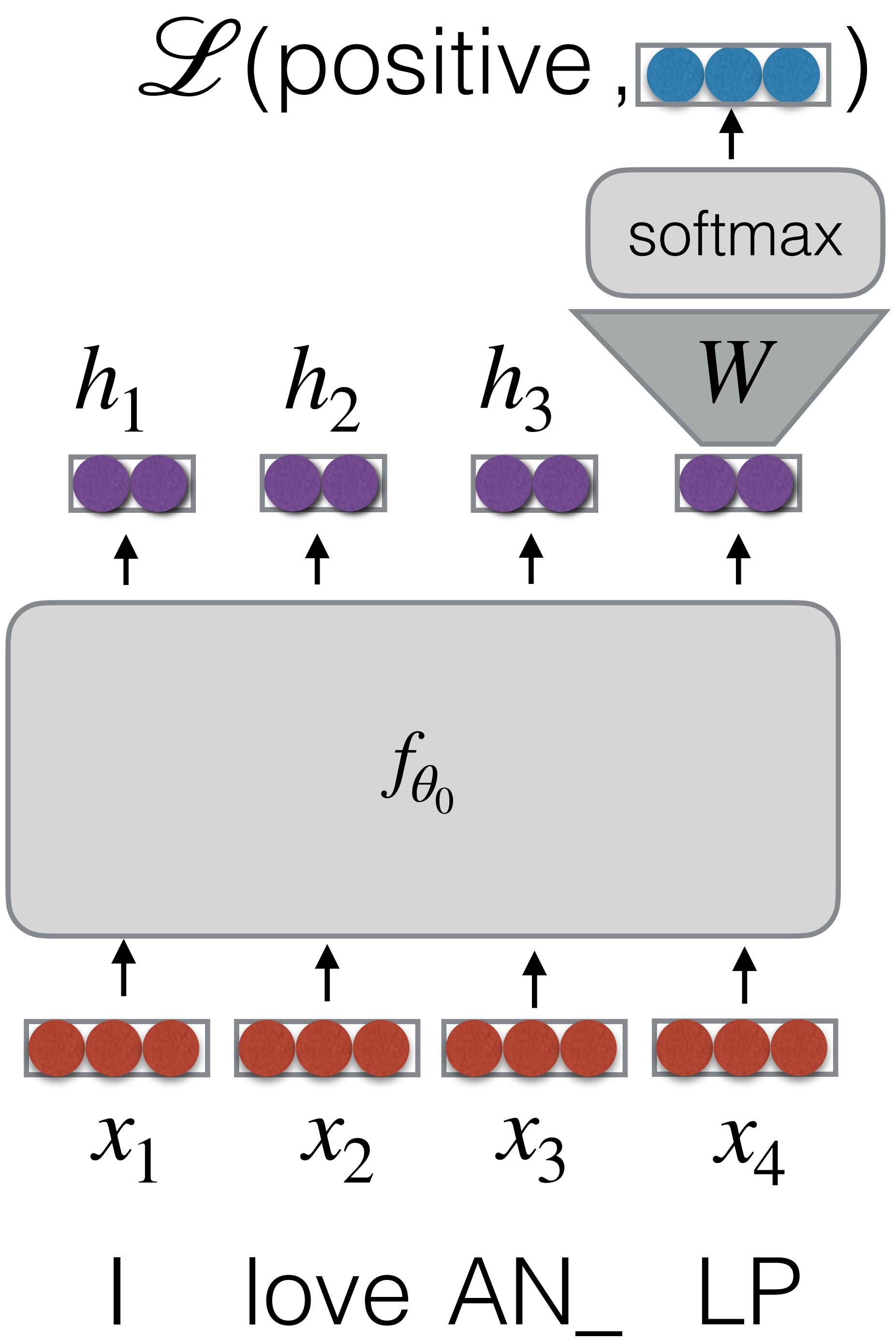
$$\theta^* = \text{argmin}_\theta \mathbb{E}_{(x,y)\sim D}\left[\mathscr{L}(f_\theta(x), y)\right]$$

+ techniques to prevent overfitting (e.g., regularization, dropout)

"Supervised fine-tuning (SFT)"

Example: classification

- Given: A sequence model $f_{\theta_0}(x_1, \ldots, x_T) \rightarrow h_1, \ldots, h_T$
  - θ_0: pre-trained weights
- Data $D = \{(x, y)_n\}_{n=1}^{N}$
  - x: input text
  - $y \in \{1,2,\ldots,K\}$ class label
- Add output head to last hidden state
  - $p_\theta(y \mid x) = \text{softmax}(Wh + b)$
    - $W \in \mathbb{R}^{K\times d}, h \in \mathbb{R}^d, b \in \mathbb{R}^K$
- Loss: cross-entropy loss $(-\log p_\theta(y \mid x))$
- By default, update all parameters $\theta = (\theta_0, W, b)$



Code example

This notebook shows fine-tuning a language model with a classification head.

Task: Given a name, predict how many vowels (a, e, i, o, u) it contains.

```
data = open
print(f"Tot
data[1000:1
```

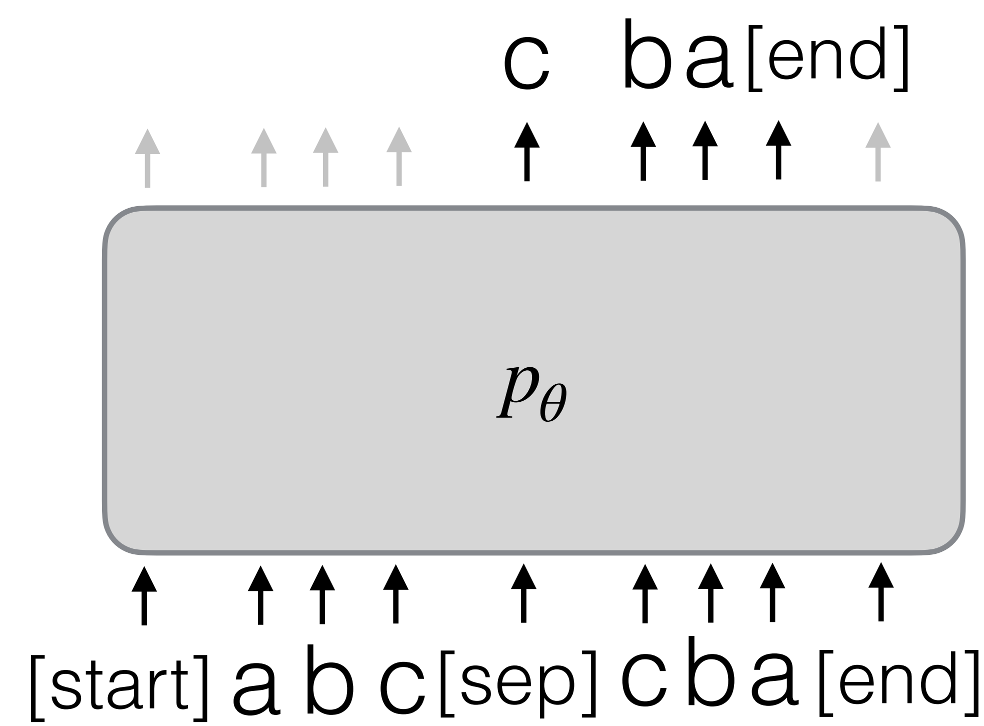
```python
class LMClassifier(nn.Module):
    def __init__(self, base_model, num_classes=11):
        super().__init__()
        self.base_model = base_model
        self.num_classes = num_classes
        self.hidden_size = base_model.config.hidden_size
        self.classifier = nn.Linear(self.hidden_size,  num_classes)

    def _last_token_hidden(self, hidden_states, attention_mask):
        # Find the position of the last non-padding token
        if attention_mask is not None:
            seq_lengths = attention_mask.sum(dim=1) - 1
            batch_size = hidden_states.size(0)
            last_hidden = hidden_states[torch.arange(batch_size), seq_lengths]
        else:
            last_hidden = hidden_states[:, -1, :]  # Last token
        return last_hidden

    def forward(self, input_ids, attention_mask=None):
        outputs = self.base_model(
            input_ids,
            attention_mask=attention_mask,
            output_hidden_states=True
        )
        hs = outputs.hidden_states[-1]  # (batch_size, seq_len, hidden_size)
        h_last = self._last_token_hidden(hs, attention_mask)
        logits = self.classifier(h_last)
        return logits
```

✓ 0.0s

Python

Language model fine-tuning

- Given: A language model $p_{\theta_0}(y \mid x)$
- Data $D = \{(x, y)_n\}_{n=1}^{N}$
  - x: input text
  - y: output text
- Loss: cross-entropy loss
  $$\mathcal{L}_{MLE} = -\sum_{t=1}^{T} \log p_\theta(y_t \mid x, y_{<t})$$



Code example

This notebook demonstrates fine-tuning a language model for a generation task.

Task: Given a name, generate its reverse (e.g., emma → amme)

```
data = open('names.txt').read().splitlines()
print(f"Total names: {len(data)}")
data[1000:1010]
```

12] ✓ 0.0s

```
Total names: 32033
```

```
Before fine-tuning:
==================================================
Reverse the name: emma. Answer: 0.4812529443310765.
Reverse the name: noah. Answer: it is only a person of that name
000B. Boole, 1
Reverse the name: olivia. Answer: a. The person is named Olivia.
```

```
After fine-tuning:
==================================================
emma        → amme      (expected: amme      ) ✓
noah        → haon      (expected: haon      ) ✓
olivia      → aivilo    (expected: aivilo    ) ✓
liam        → mail      (expected: mail      ) ✓
sophia      → aihpos    (expected: aihpos    ) ✓
mason       → nosam     (expected: nosam     ) ✓
isabella    → allebasi  (expected: allebasi  ) ✓
william     → luottiw   (expected: mailliw   ) ✗
mia         → aim       (expected: aim       ) ✓
james       → semaj     (expected: semaj     ) ✓
```

Library 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

Should I fine-tune all parameters?

- Option 1: Update only the output head (W, b)
  - Cheap: only $K \times d + K$ parameters
  - Assumes that the pre-trained representations are good, e.g. linearly separate the labels
- Option 2: Update all parameters
  - Expensive: $|\theta|$ parameters (e.g., 1M, 1B, 100B, …)
  - Changes the representations
  - May lead to overfitting
- Option 3: Update a small number of parameters inside the model
  - Cheaper: $<< |\theta|$ parameters
  - Can change the representations
  - “Parameter-efficient fine-tuning (PEFT)” methods

Example: Low-Rank Adaptation (LoRA)

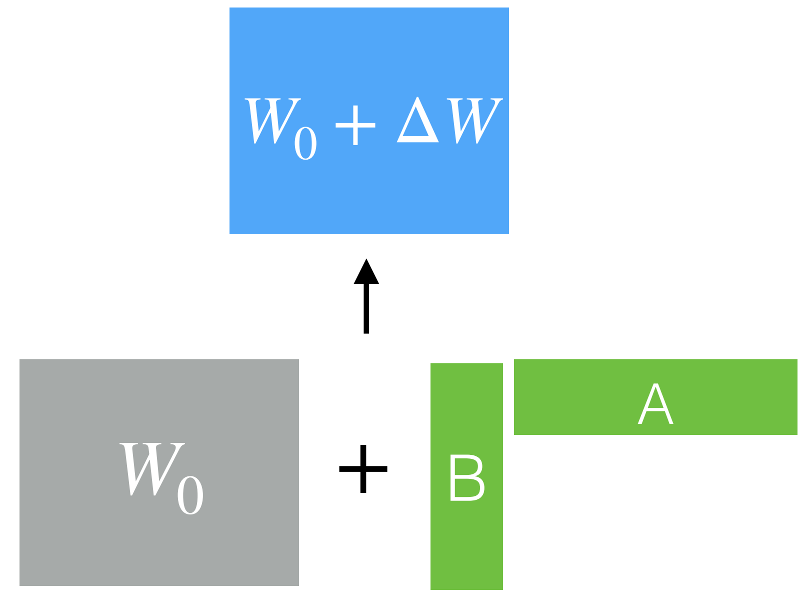
[Hu et al 2021]

- Given weights $W_0 \in \mathbb{R}^{d \times d'}$, introduce new weights A, B:

$$W_0 + \underbrace{BA}_{\Delta W}$$

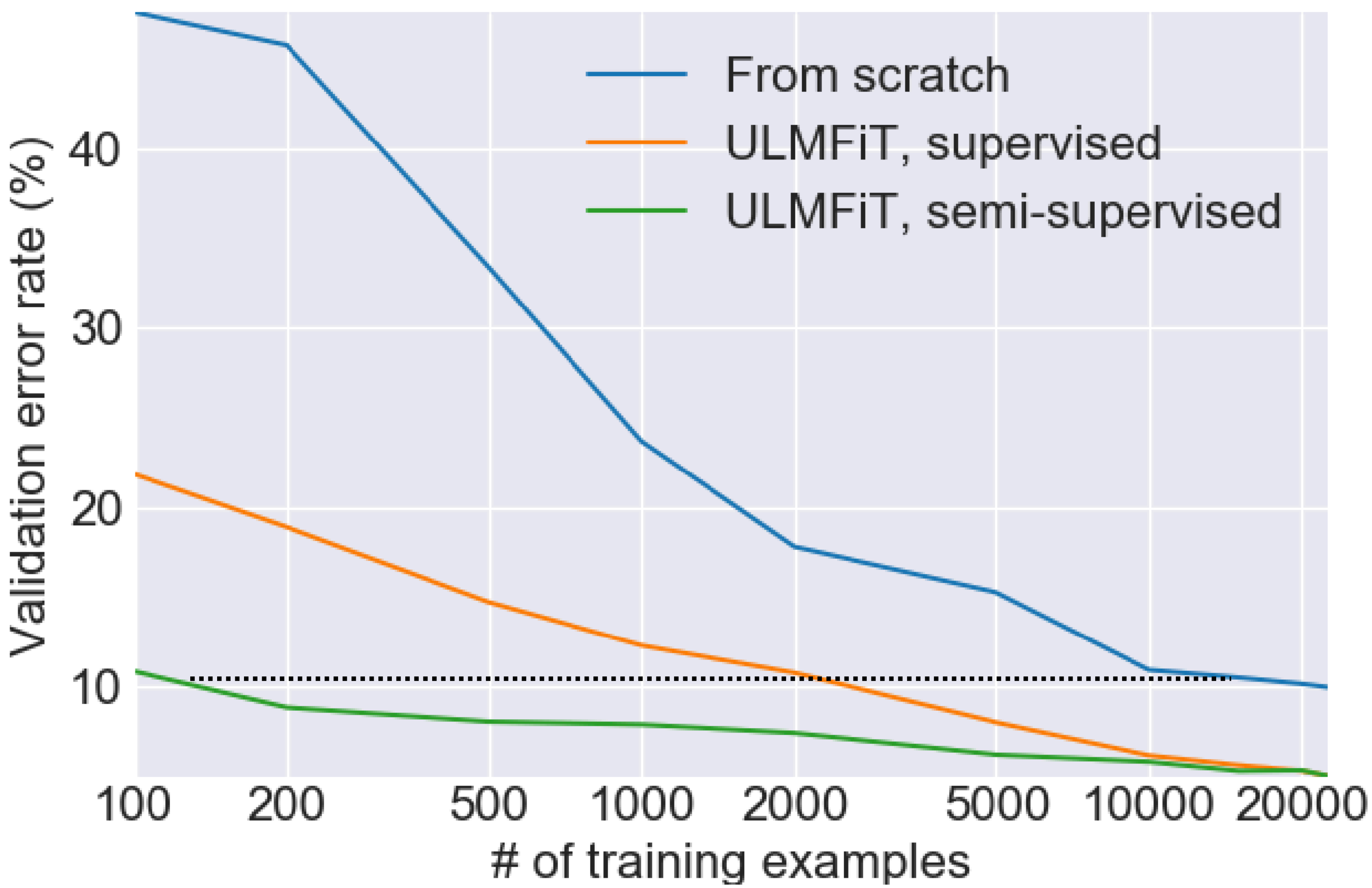
where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times d'}$, $r \ll \min(d, k)$

- *Only* update B and A during fine-tuning.
- After fine-tuning, simply use the weight matrix $W = W_0 + \Delta W$
- Scale ΔW by $\dfrac{\alpha}{r}$
- Apply to W_q and W_v in the attention layers



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

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 …

```
Testing different prompts for 'sophia' (expected: aihpos)
============================================================
Train     :
  Prompt: Reverse the name: sophia. Answer:...
  Output: aihpos     ✓
Variation 1:
  Prompt: Reverse: sophia =>...
  Output: aihpos     ✓
Variation 2:
  Prompt: What is sophia backwards?...
  Output:            ✗
Variation 3:
  Prompt: sophia reversed is:...
  Output: aihsopi    ✗
```

Example data

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

Example data

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

Example data

X	Y
Article (paper)	Abstract
Problem	Solution
Prompt	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

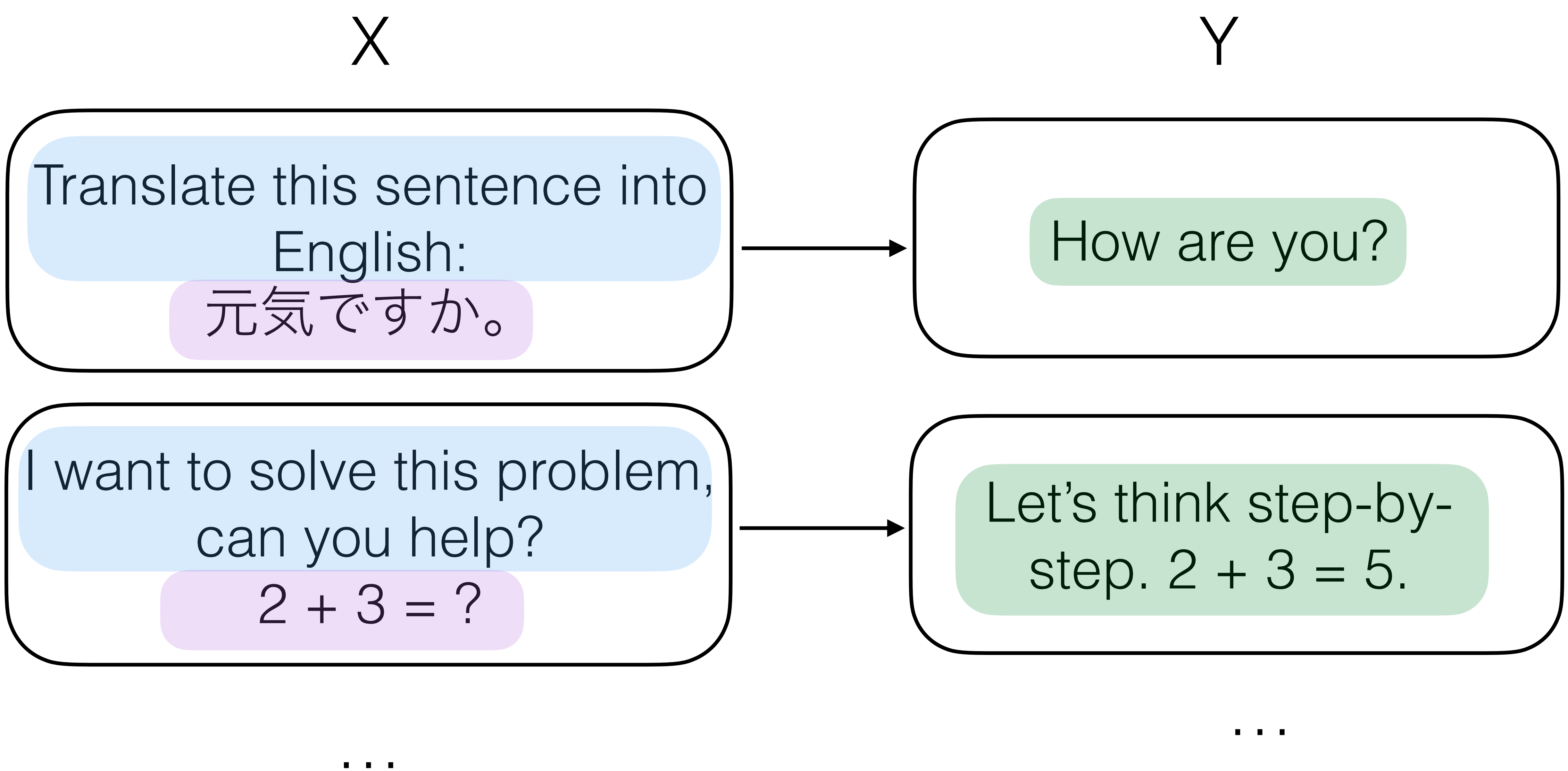
https://huggingface.co/datasets/Open-Orca/OpenOrca

Today's lecture

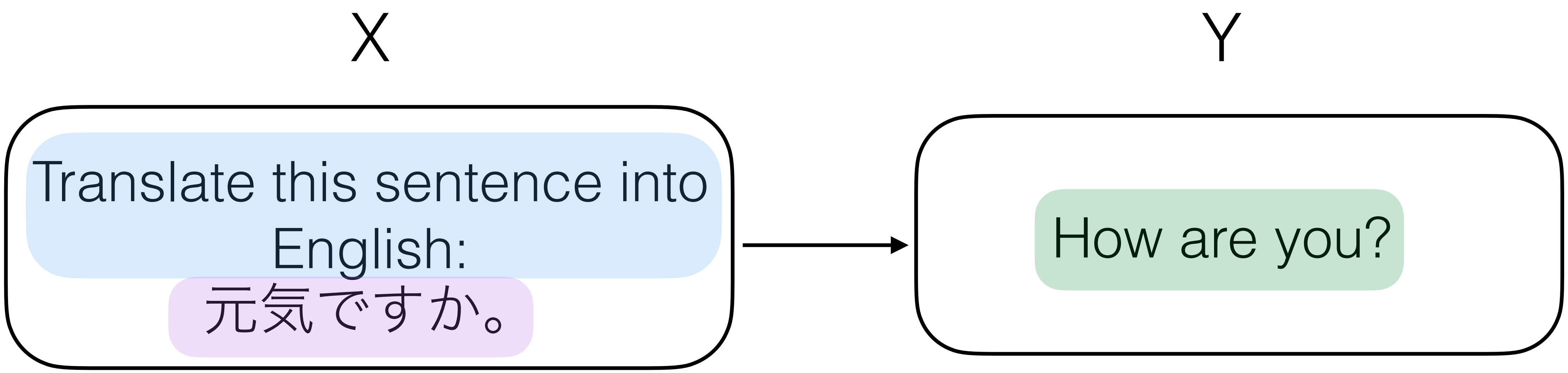
- Fine-tuning basics
- **Instruction tuning**
    - Chat tuning

Basic idea

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



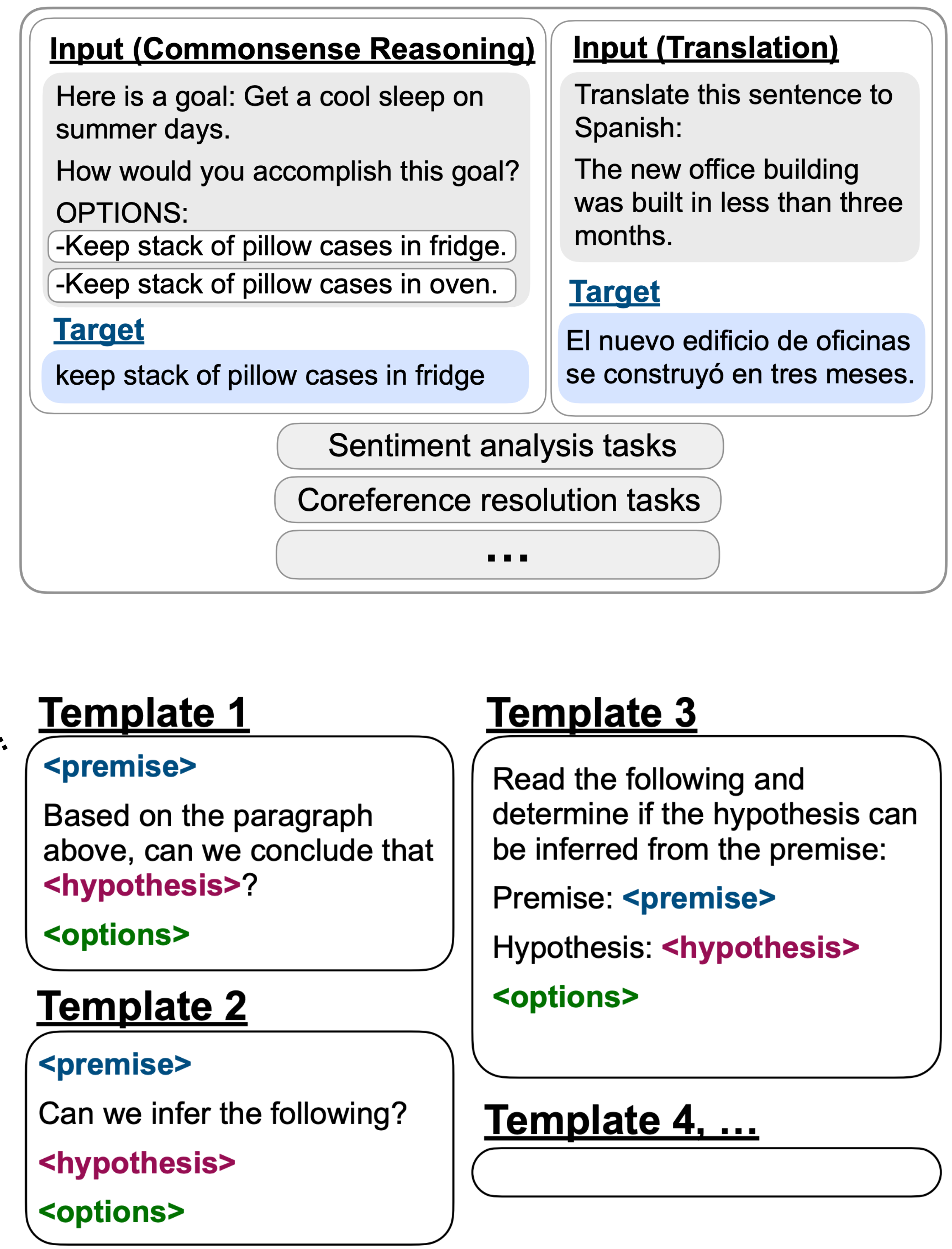
Variations



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

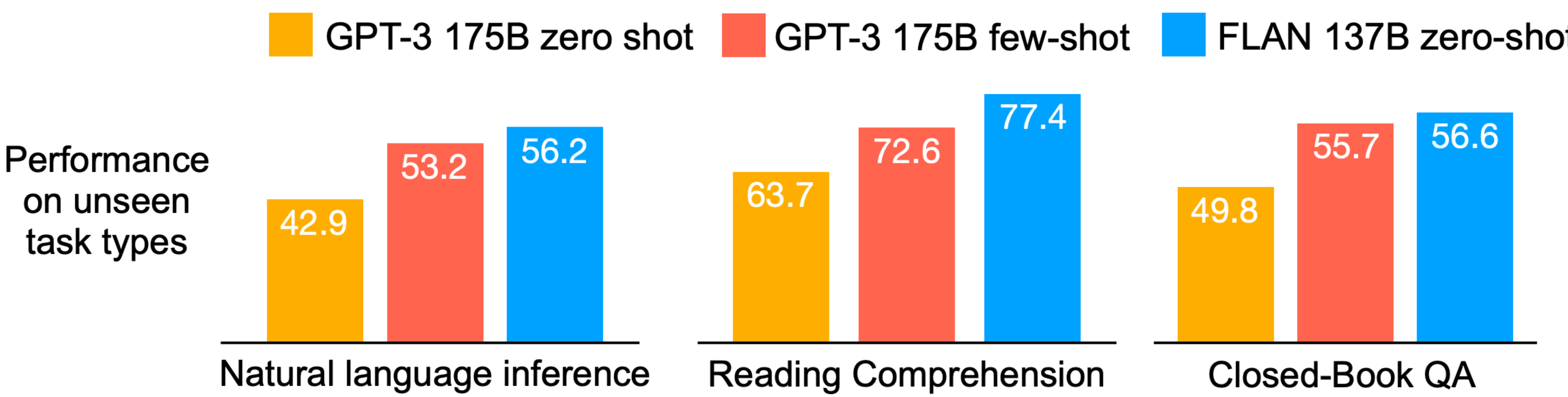
Example: FLAN [Wei et al 2021]

- 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

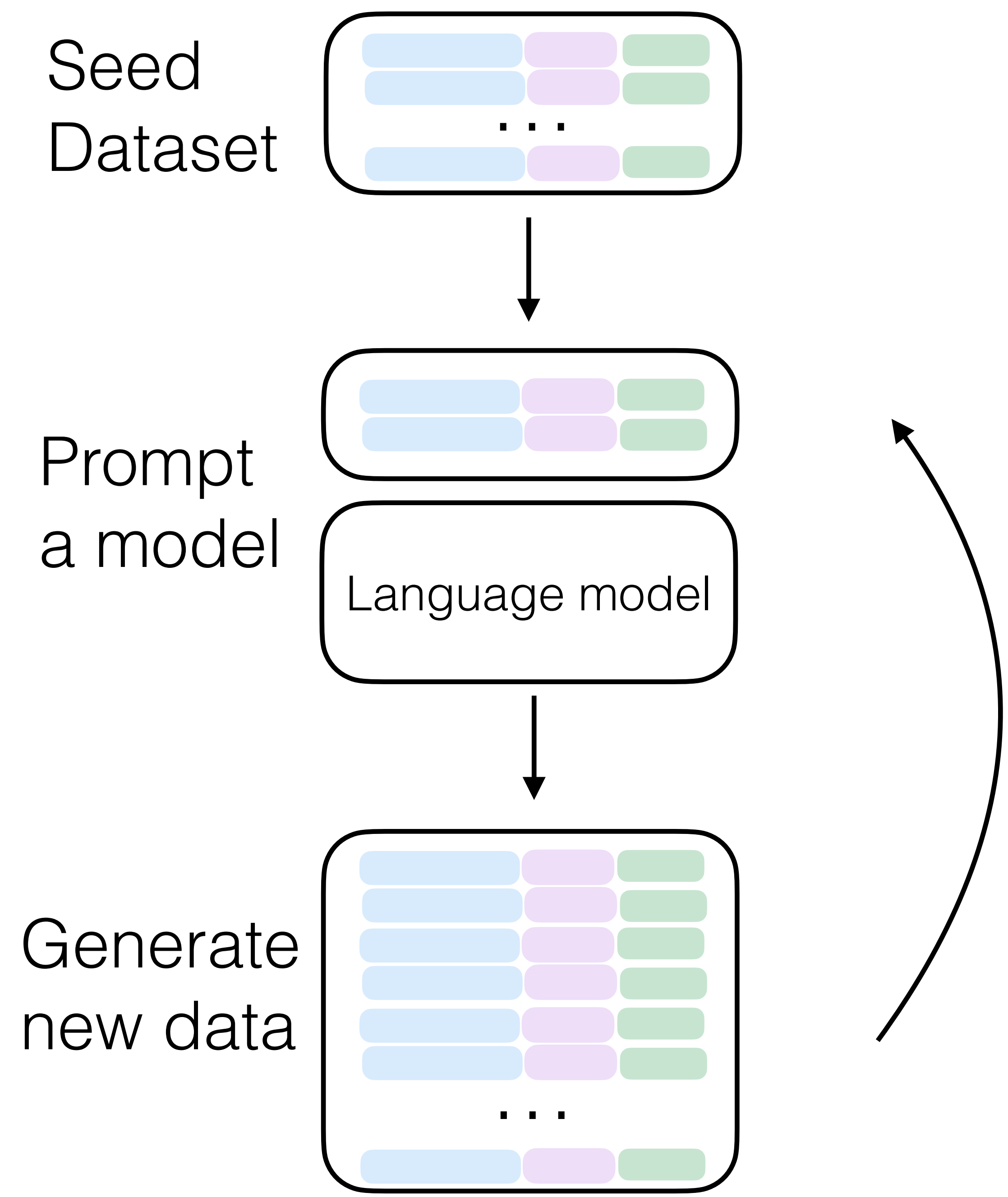
[Mishra et al 2021, Wang & Mishra et al 2022]

- 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 7

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 | multi-turn

- Example: LMSys-1M

conversation list · *lengths* 2↔45 99.8%	turn int64 1↔22 99.8%
[{ "content": "Beside OFAC's selective sanction that target the listed individiuals and entities, please elaborate on the other types of US's sanctions, for example, comprehensive and sectoral sanctions. Please be detailed as much as possible", "role": "user" }, { "content": "The United States has a number of different types of sanctions that it can use to achieve its foreign policy goals, including both selective and comprehensive sanctions.\n\nSelective sanctions are targeted at specific individuals or entities that are believed to be engaged in activities that are contrary to US interests. These sanctions can take a variety of forms, including asset freezes, travel bans, and restrictions on financial transactions. The Office of Foreign Assets Control	6

Data source: online LLM service hosted by Berkeley/Stanford

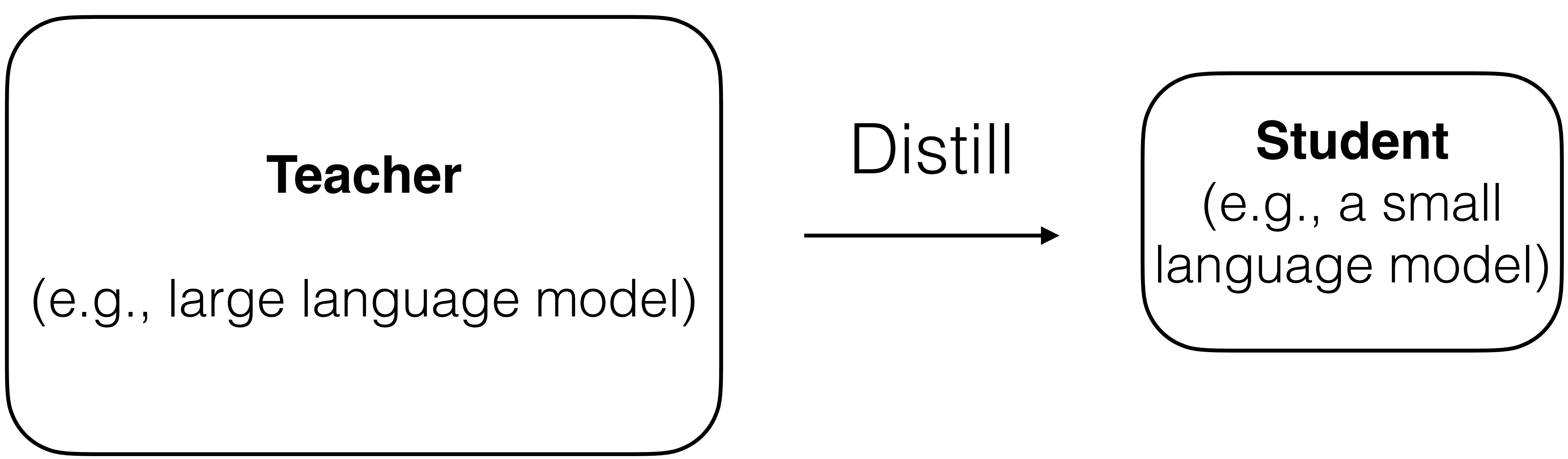
https://huggingface.co/datasets/lmsys/lmsys-chat-1m

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]



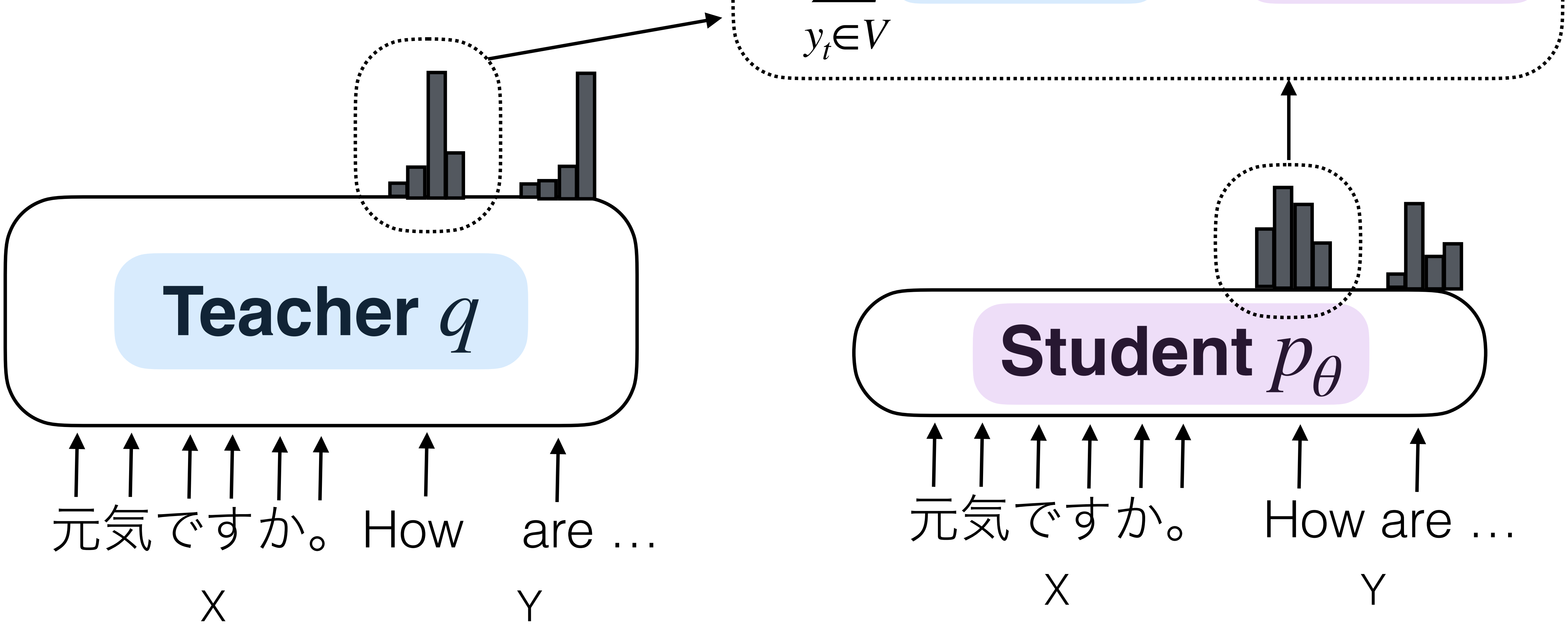
Token-level knowledge distillation

[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:

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

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

"Soft labels"

Sequence-level knowledge distillation

[Kim & Rush 2016]

- Generate with a teacher model

Prompt 1 → **Teacher q** $\xrightarrow{\text{Generate}}$ Response 1

Prompt 2 Response 2

… …

- Student model fine-tunes on the generated data

(Prompt, Response) dataset $\xrightarrow{\text{Fine-tune}}$ **Student p_θ**

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

Sequence-level knowledge distillation

[Kim & Rush 2016]

- Also minimizes KL between teacher and student:

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

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

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

Recap

- Fine-tuning basics
  - Adjust a model's parameters using data
  - PEFT: only update a small number of parameters
- Instruction tuning
  - Format data so that a model learns to do multiple tasks
- Knowledge distillation
  - Data can come from various teachers (human, model)

Thank you