CS 1671 / CS 2071
Human Language Technologies

Session 21: Prompting, finetuning, and evaluating LLMs

Michael Miller Yoder

November 3, 2026



School of Computing and Information

Assessments: quiz

(Final) quiz in class **this Thu Nov 5** covering:

- Session 19 Post-training LLMs
  - J+M 8
- Session 20 BERT
  - J+M 9-9.3, 9.5-9.5.1, 9.6
- Session 21 Prompting, finetuning, and evaluating LLMs
  - J+M 1.9, 8.2

Assessments: exam

- Exam is in class next **Tue Nov 10**
- Covers everything in the class up to that point
- In-class, paper exam
- Closed-book except for 1 letter page of double-sided notes
- Conceptual
- No programming
- Class this Thu Nov 5 will be a review session for the exam

Assessments: homework

- Homework 3 is **due Nov 17**
- Run Jupyter notebooks from templates on the CRCD
- Part 1: LLM prompting
- Part 2: Instruction tuning of an LLM
  - CRCD GPUs

Overview: Prompting, finetuning, and evaluating LLMs

- Coding activity: finetuning BERT for politeness classification
- LLM discussion
- Prompting
  - In-context learning
  - Chain-of-thought prompting
- Parameter-efficient finetuning with LoRA
- Evaluation of text generation: automatic proxy metrics
- Review of types of LLM finetuning

CRCD GPUs for Jupyter notebooks on the teach cluster

GPUs from most to least powerful:

1. Nvidia L4 (16 available)
2. Nvidia GTX Titan X (28 available)
3. Nvidia GTX 1080 (32-78 available)

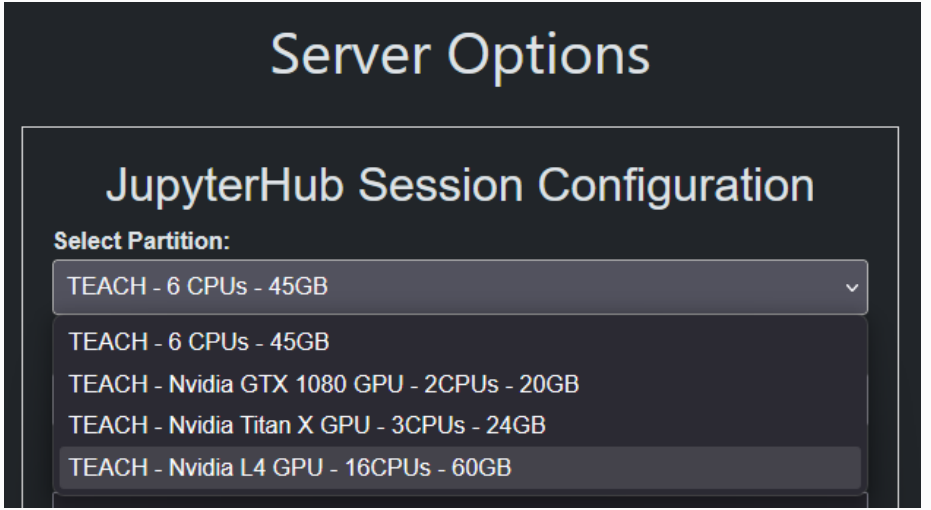
If you get a “Pending in queue…” message, all GPUs of that type may be in use. **Try another type.**



Check the status of the L4 servers by running **sinfo –M teach** from a CRCD ssh session or a JupyterHub terminal and looking at nodes 15-16

Notebook: finetune BERT for politeness classification

1. Click this nbgitpuller link (also available on course website)
2. **Use a GPU:** Start a server with ‘TEACH – Nvidia L4 GPU – 16 CPUs – 60GB’ server
3. Load **new** custom environment at /vast/cs1671-2026f/class-env/.pixi/envs/default
4. Open session20_bert_politeness.ipynb



Discussion: LLMs

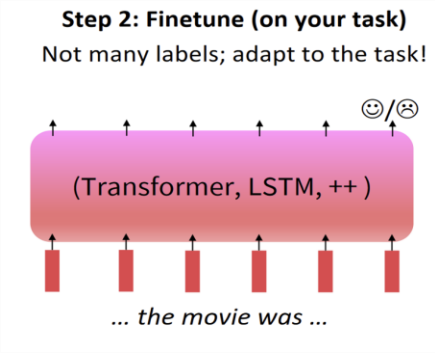
Now that we've gone over how different types of transformer-based LLMs are pretrained and post-trained,

1. Do you think these models understand human language? Do you think these models can reason in complicated scenarios?
2. What prior technologies have had a similar widespread social impact in information access? What lessons can we learn from how those technologies were incorporated into social life?
3. When should we be careful when using LLMs? In other words, what kind of tasks are LLMs good at, and what kind of tasks are LLMs bad at?

In-context learning, zero-shot and few-shot learning

Two ways of adapting an LLM to your use case

1. Finetune the parameters of the model with additional data
    - See parameter-efficient finetuning for finetuning very large models



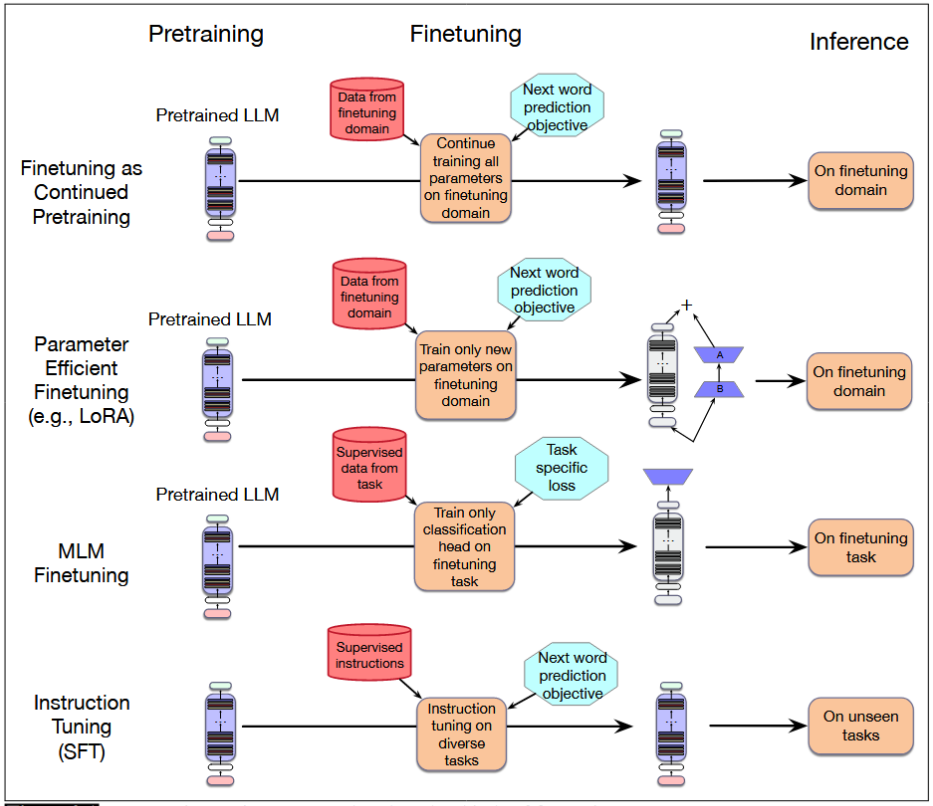


Figure 9.1 Instruction tuning compared to the other kinds of finetuning.

Figures from John Hewitt, Jurafsky and Martin

Two ways of adapting an LLM to your use case

2. Add more information in your prompt to the LLM, such as demonstrations (in-context learning)

Emergent zero-shot learning from GPT-2

- GPT-2 [Radford et al. 2019] is a 1.5B parameter language model trained on 40G text data (webtext)
- One key emergent ability in GPT-2 is zero-shot learning: the ability to do many tasks with **no examples** and **no gradient updates**, by simply:
  - Specifying the right sequence prediction problem in the prompt
    - Question answering

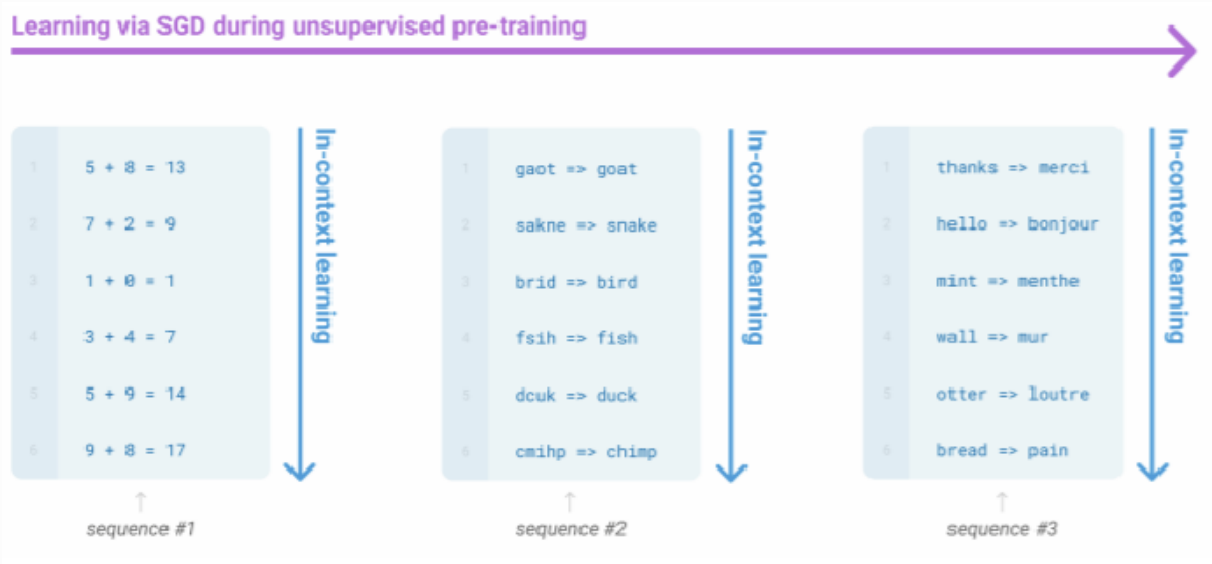
      ```
      Passage: Tom Brady... Q: Where was Tom Brady born? A: ...
      ```

    - Summarization: <article> tl;dr <summary>

Slide adapted from Stanford Natural Language Processing with Deep Learning course via Lorraine Li

In-context learning (few-shot prompting)

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts (prompts). The in-context steps seem to specify the task to be performed.



Slide adapted from John Hewitt

Chain-of-thought prompting

Limits of prompting for harder tasks?

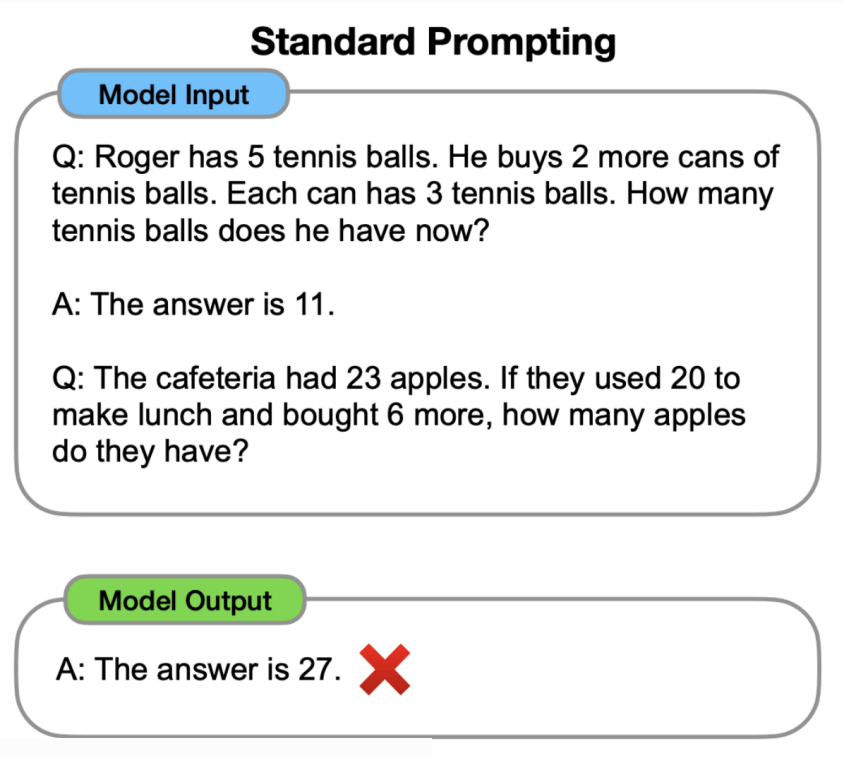
- Some tasks seem too hard for LLM to learn through prompting alone.
- Especially tasks involving richer, multi-step reasoning

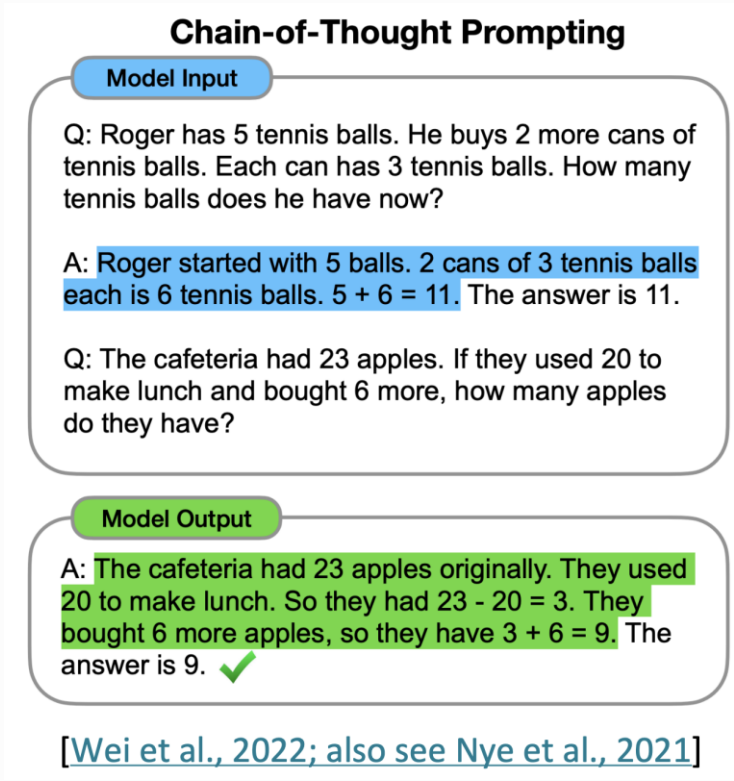
```
19583 + 29534 = 49117
98394 + 49384 = 147778
29382 + 12347 = 41729
93847 + 39299 = ?
```

Solution: change the prompt!

Slide adapted from Stanford Natural Language Processing with Deep Learning course

Chain-of-thought prompting

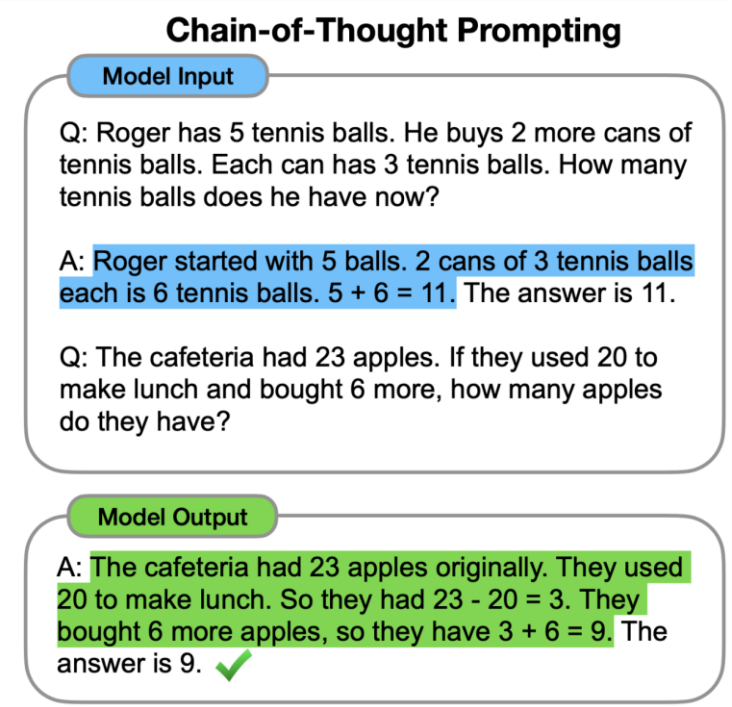




[Wei et al., 2022; also see Nye et al., 2021]

Slide adapted from Stanford Natural Language Processing with Deep Learning course via Lorraine Li

Chain-of-thought prompting



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

Slide adapted from Stanford Natural Language Processing with Deep Learning course via Lorraine Li

Chain-of-thought prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

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

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

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

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

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

[Kojima et al., 2022]

Slide adapted from Stanford Natural Language Processing with Deep Learning course via Lorraine Li

Downside of prompt-based learning

1. **Inefficiency:** The prompt needs to be processed *every time* the model makes a prediction.

2. **Poor performance**: Prompting generally performs worse than fine-tuning [Brown et al., 2020].

3. **Sensitivity** to the wording of the prompt [Webson & Pavlick, 2022], order of examples [Zhao et al., 2021; Lu et al., 2022], etc.

4. **Lack of clarity** regarding what the model learns from the prompt. Even random labels work [Zhang et al., 2022; Min et al., 2022]!

Slide adapted from Stanford Natural Language Processing with Deep Learning course via Lorraine Li

Parameter-efficient finetuning

Parameter-Efficient Finetuning

Adapting to a new domain by continued pretraining (finetuning) is a problem with huge LLMs.

- Enormous numbers of parameters to train
- Each pass of batch gradient descent has to backpropagate through many, many huge layers.
- Expensive in processing power, in memory, and in time.

Instead, **parameter-efficient fine tuning (PEFT)**

- Efficiently select a subset of parameters to update when finetuning.
- Freeze some of the parameters (don't change them),
- And only update some a few parameters.

Slide adapted from Jurafsky and Martin

LoRA (Low-Rank Adaptation)

Transformers have many dense matrix multiplication layers

- Like W^Q, W^K, W^V, W^O layers in attention

Instead of updating these layers during finetuning,

- Freeze these layers
- Update a low-rank approximation with fewer parameters.

Slide adapted from Jurafsky and Martin

LoRA

Consider a **matrix W** (shape [$k \times d$]) that needs to be updated during finetuning via gradient descent.

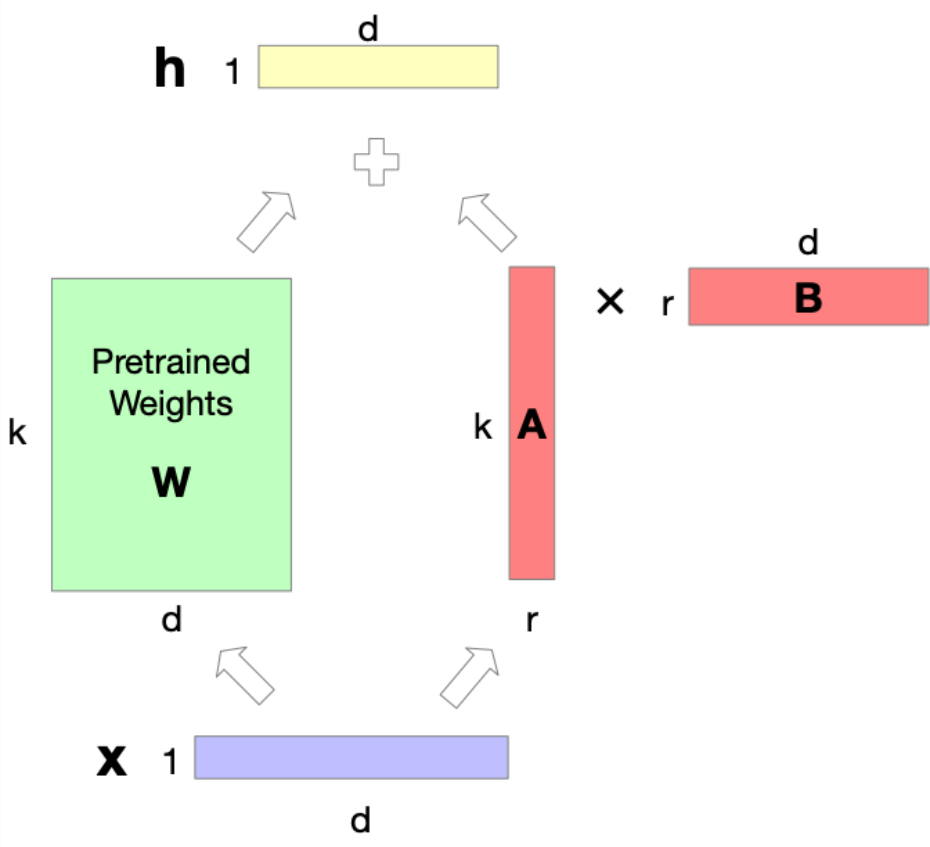
- Normally updates are ΔW (shape [$k \times d$])

In LoRA, we freeze W and update instead a low-rank decomposition of W:

- A of shape [$k \times r$],
- B of shape [$r \times d$],
- r is very small (like 1 or 2)
- Finetuning updates A and B instead of W.

Forward pass: instead of **h = xW**

We do **h = xW + xAB**



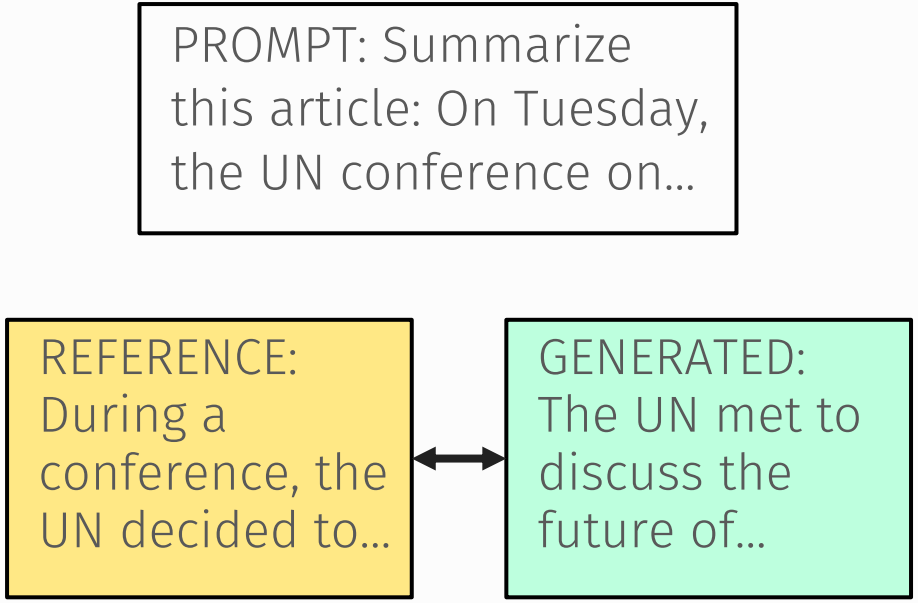
Slide adapted from Jurafsky and Martin

Evaluating generated text

How do we evaluate LLMs?

- Perplexity
  - Which model assigns higher probability to an unseen test set of language
- On ability on downstream tasks:
  - Answering multiple-choice questions correctly (MMLU dataset) or do things like pass the bar exam
- For specific tasks, on their ability to “match” a correct reference text

Evaluation metrics for matching texts (proxy metrics)



- LLM-as-a-judge
  - Pass generated text and reference text to another LLM to judge how similar they are
- ROUGE
  - Measures n-gram overlap
- BERTScore
  - Measures the distance between BERT embeddings of the system output and the reference
  - Attempts to handle synonymy

Review: types of finetuning

For each type:

1. What is the format of the type of data used for finetuning?
2. Anything else that distinguishes it from the others?

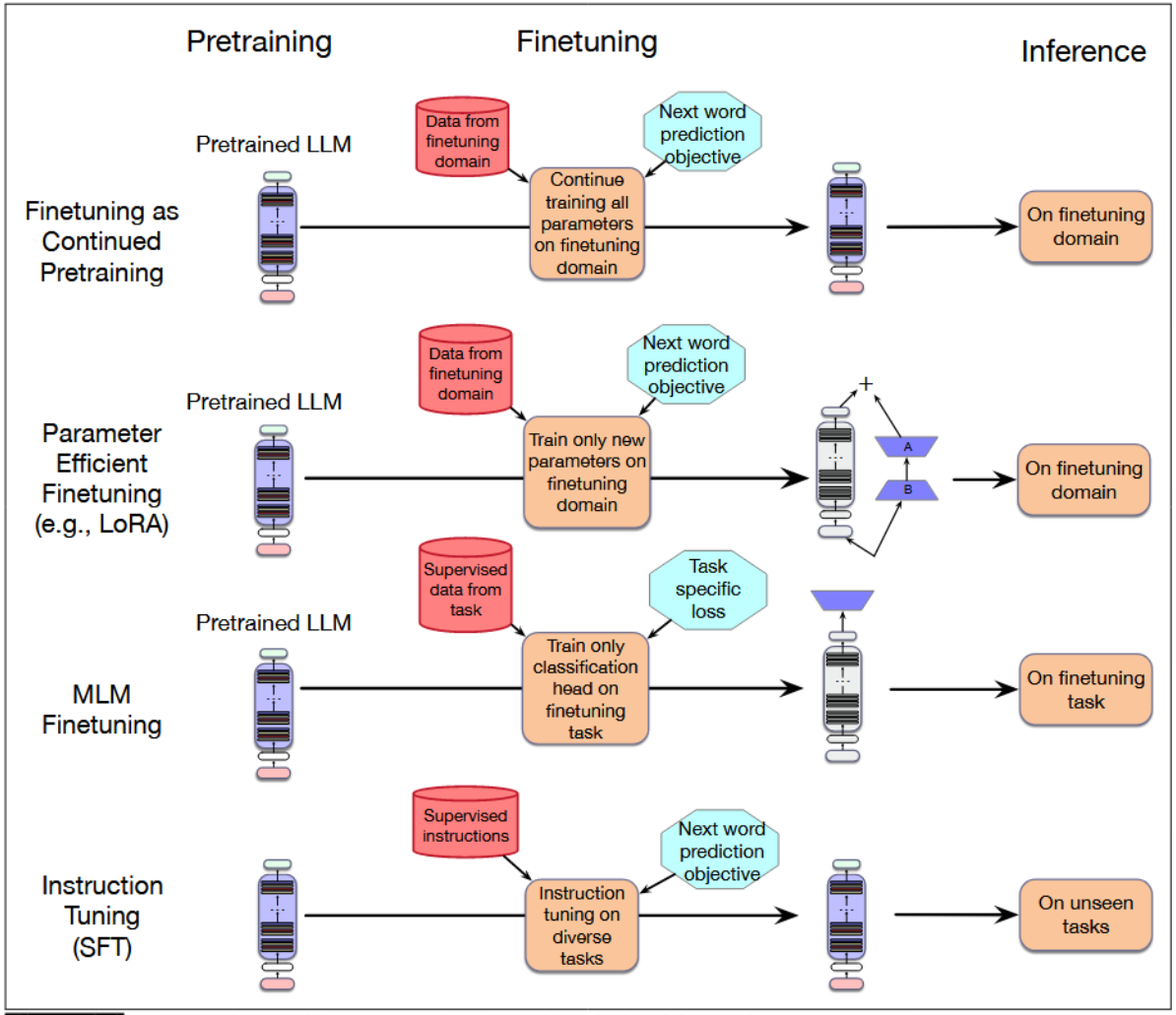


Figure 9.1 Instruction tuning compared to the other kinds of finetuning.

Figure from Jurafsky and Martin

Conclusion

- Zero-shot prompting is simply asking an LLM to do something without providing any examples
- Few-shot prompting (in-context learning) is where a few examples are provided, which can improve LLM output
- Chain-of-though prompting, providing reasoning in examples, can also improve LLM output
- Parameter-efficient finetuning methods such as LoRA only update a small set of weights during finetuning
- ROUGE and BERTScore are two evaluation metrics for testing the difference between a reference text and a system-generated text