



Instruction Tuning

LLMs Series

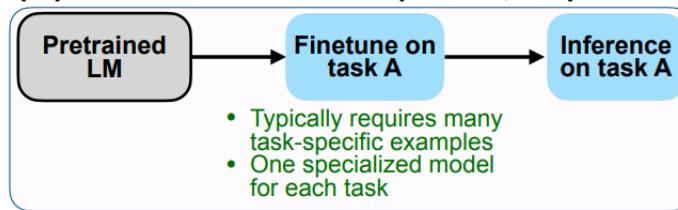
Dinh-Thang Duong – TA

Yen-Linh Vu – STA

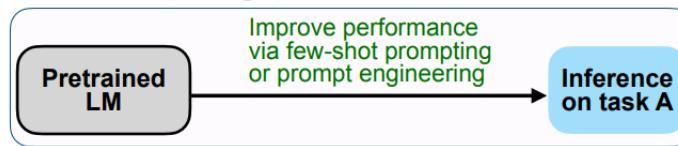
Anh-Khoi Nguyen – STA

Objectives

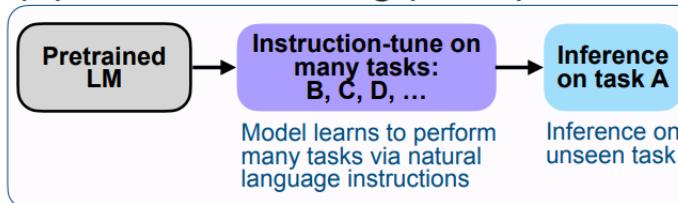
(A) Pretrain–finetune (BERT, T5)



(B) Prompting (GPT-3)



(C) Instruction tuning (FLAN)



<|im_start|> system
Bạn là một chuyên gia về toán. Bạn sẽ nhận câu hỏi trắc nghiệm kèm theo các lựa chọn, hãy giải step by step nếu có và chọn phương án đúng.

<|im_start|> user
Câu hỏi:
Điền dấu >, <, = vào ô trống: (13 - 3) ... 3
Các lựa chọn:
A. <
B. >
C. =
Câu trả lời:

<|im_start|> assistant
B. >

Giải thích:
Trong phép tính $(13 - 3)$, kết quả là 10 . Vậy giờ chúng ta trừ 3 từ 10 , kết quả là 7 . Vì 7 lớn hơn 3 , chúng ta điền $>$ vào ô trống.
CPU times: user 8.34 s, sys: 132 ms, total: 8.47 s
Wall time: 14.1 s

In this study session, we will discuss about:

1. How LLMs learn (training type)?
2. How to make LLMs adapt to a task?
 1. Prompting (In-context learning).
 2. Instruction Tuning.
3. How to train LLMs on a single (small) GPU?
4. Apply Instruction Tuning for multiple choice SAT English Reading.

Outline

- Introduction
- In-context Learning
- Supervised Fine-tuning
- Instruction Tuning
- Deployment
- Question

 AI

AI VIET NAM
@aivietnam.edu.vn

Introduction

Introduction

❖ What are Large Language Models (LLMs)?

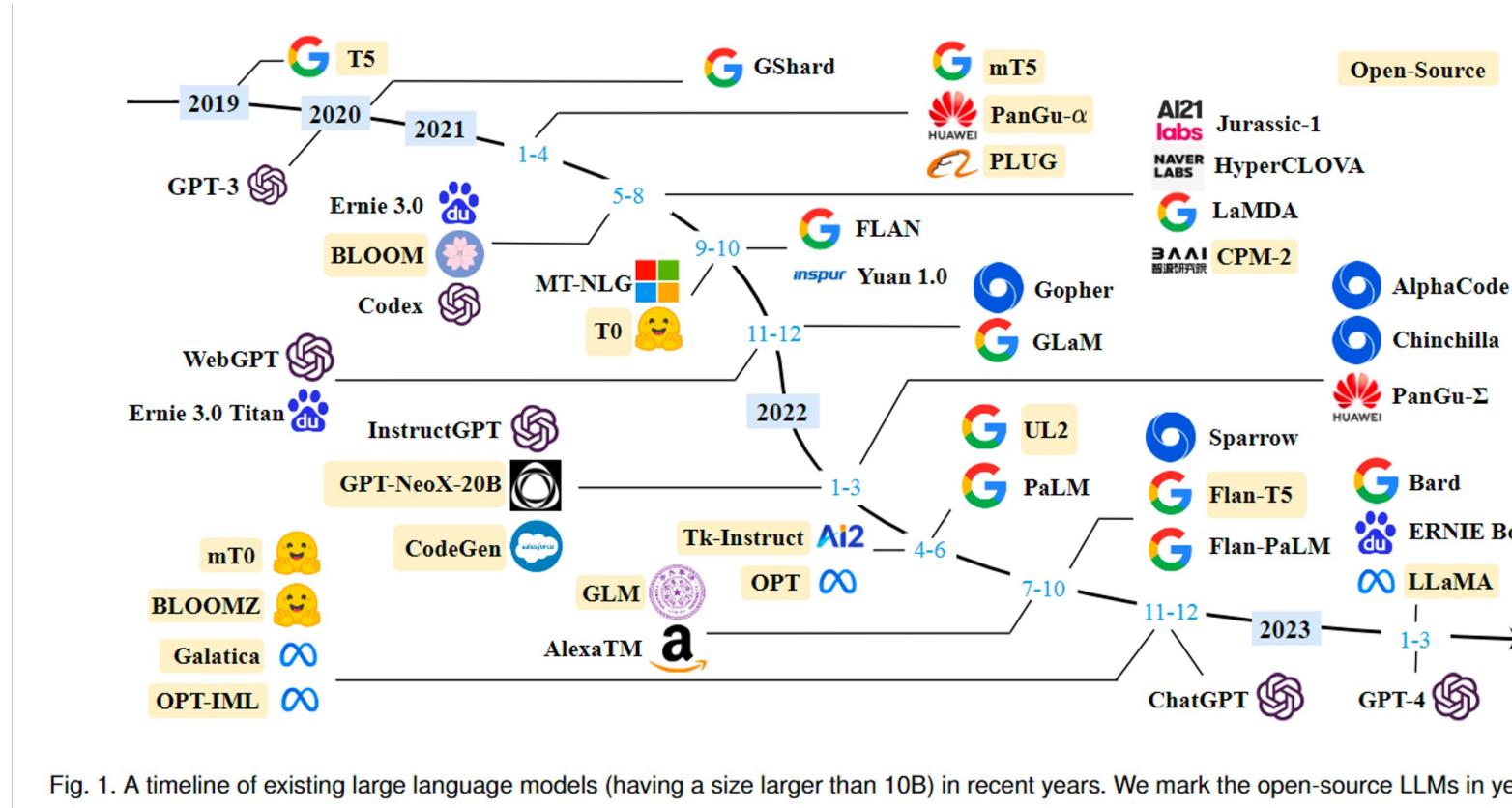


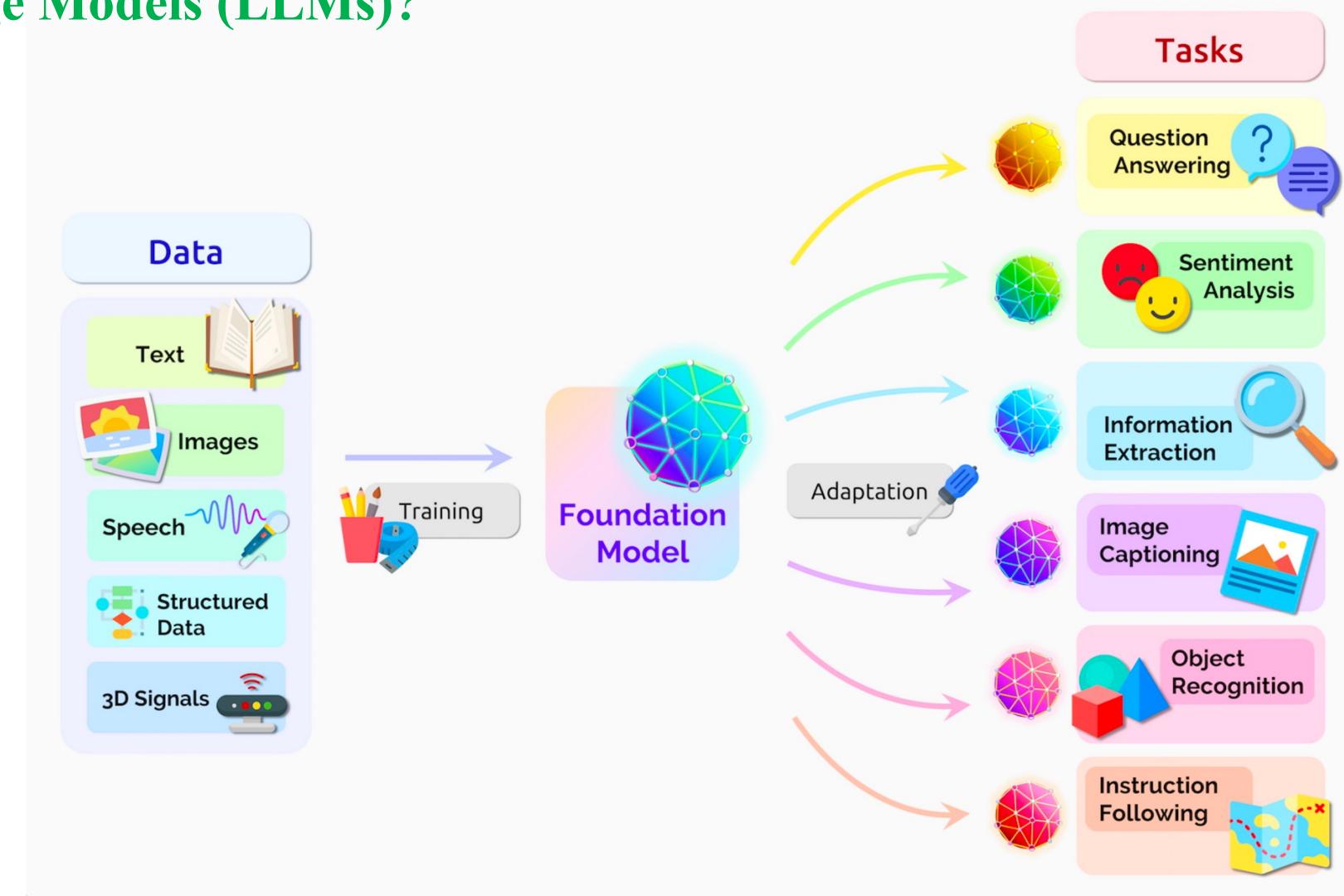
Fig. 1. A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open-source LLMs in yellow color.

LLMs (Large Language Models): AI models (language models) that were trained on a very large corpus of text. This made them capable of performing various NLP tasks with high precision.

Introduction

❖ What are Large Language Models (LLMs)?

LLMs are often pretrained on a vast majority of data and designed to be adaptable to a wide variety of tasks (Foundation models).



Introduction

❖ Getting Started

Textual Description (Prompt):

Write a python function that receive an image and plot it using matplotlib library.

 You
Write a python function that receive an image and plot it using matplotlib library.

 ChatGPT
To create a Python function that receives an image and plots it using the Matplotlib library, you can use the following code as an example. You'll need to have Matplotlib and a library like PIL (Pillow) installed to work with images:

```
python
import matplotlib.pyplot as plt
from PIL import Image

def plot_image(image_path):
    try:
        # Open the image using PIL (Pillow)
        img = Image.open(image_path)

        # Create a Matplotlib figure and axis
        fig, ax = plt.subplots()
```

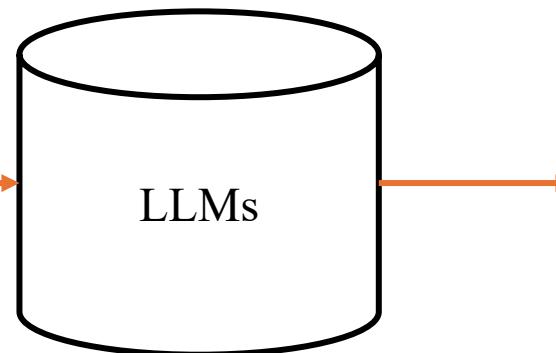
 Copy code

Introduction

❖ LLMs I/O

Input Text

Write a python function that receive an image and plot it using matplotlib library.



Given a “prompt”, LLMs can generate an appropriate response.

Output Text

AI You Write a python function that receive an image and plot it using matplotlib library.

ChatGPT To create a Python function that receives an image and plots it using the Matplotlib library, you can use the following code as an example. You'll need to have Matplotlib and a library like PIL (Pillow) installed to work with images:

```
python
import matplotlib.pyplot as plt
from PIL import Image

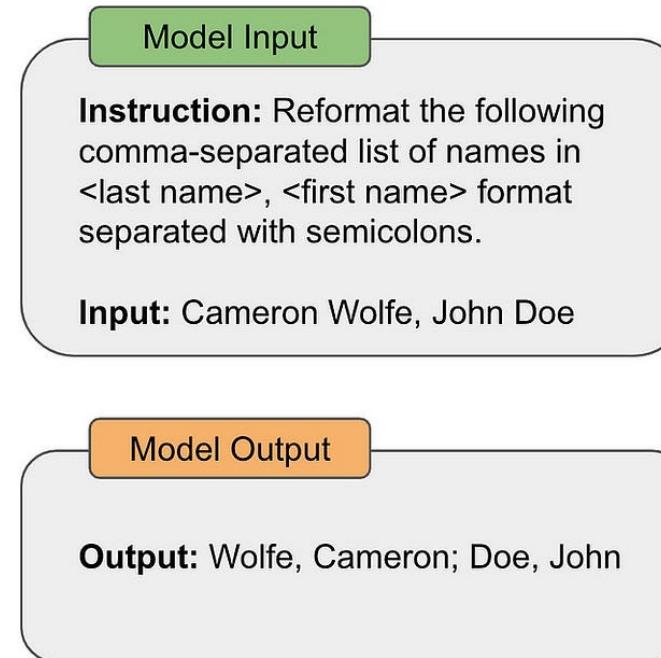
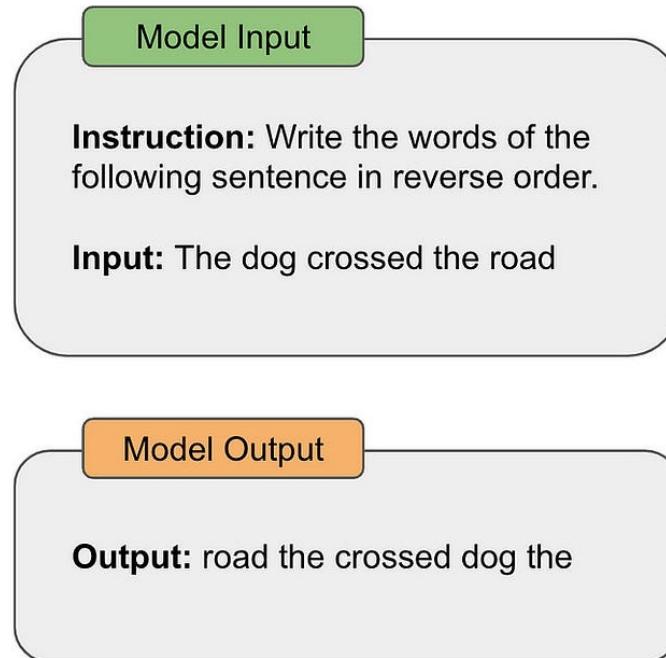
def plot_image(image_path):
    try:
        # Open the image using PIL (Pillow)
        img = Image.open(image_path)

        # Create a Matplotlib figure and axis
        fig, ax = plt.subplots()
```

Copy code

Introduction

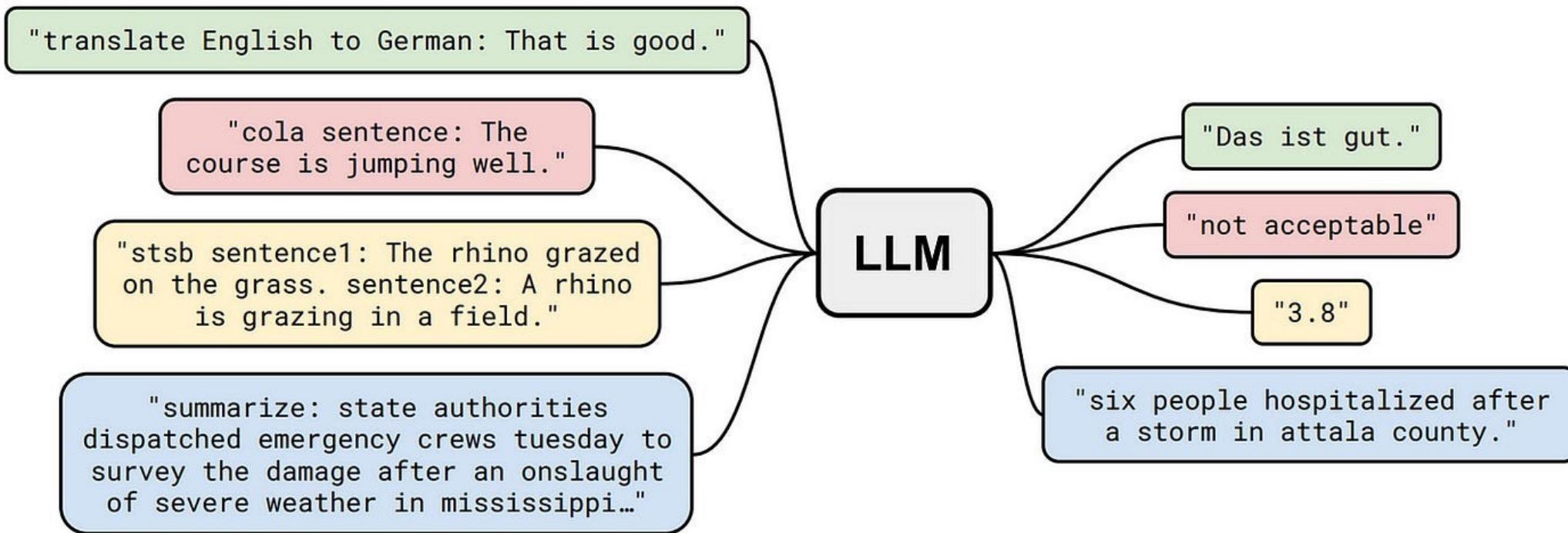
❖ Generative AI Prompting



Prompting: Refers to a process of providing an input, usually in the form of text data, to a generative AI model to generate a specific output.

Introduction

❖ Prompting in LLMs



With prompting, we can make LLMs do any task with just natural language (**zero-shot capability**).

Introduction

❖ Getting Started

<|im_start|> system

Bạn là một trợ lí AI hữu ích. Hãy trả lời người dùng một cách chính xác.

<|im_start|> user

Mô tả về thành phố Đà Lạt

<|im_start|> assistant

Đà Lạt là một thành phố đẹp như tranh vẽ nằm ở vùng cao nguyên của tỉnh Lâm Đồng, Việt Nam. Được biết đến như "Thành phố ngàn hoa", Đà Lạt nổi tiếng với những khu vườn tươi tốt, những con đường đẹp như tranh vẽ và những ngôi nhà đầy màu sắc. Thành phố này có khí hậu mát mẻ, với nhiệt độ trung bình từ 15 đến 20 độ C vào những tháng mùa đông. Vào những tháng mùa hè, nhiệt độ có thể lên tới 25 độ C. Đà Lạt nổi tiếng với các món ăn ngon, từ các món ăn truyền thống Việt Nam đến các món ăn quốc tế. Thành phố cũng nổi tiếng với các lễ hội và sự kiện văn hóa, chẳng hạn như Lễ hội hoa hồng, Lễ hội mùa đông và Lễ hội mùa xuân. Đà Lạt là một điểm đến tuyệt vời cho những người yêu thích thiên nhiên, những người đam mê ẩm thực và những người tìm kiếm một nơi yên bình để thư giãn.

Example: AI Chatbot using Large Language Models (LLMs).

Introduction

❖ LLMs problem



$241 - (-241) + 1$



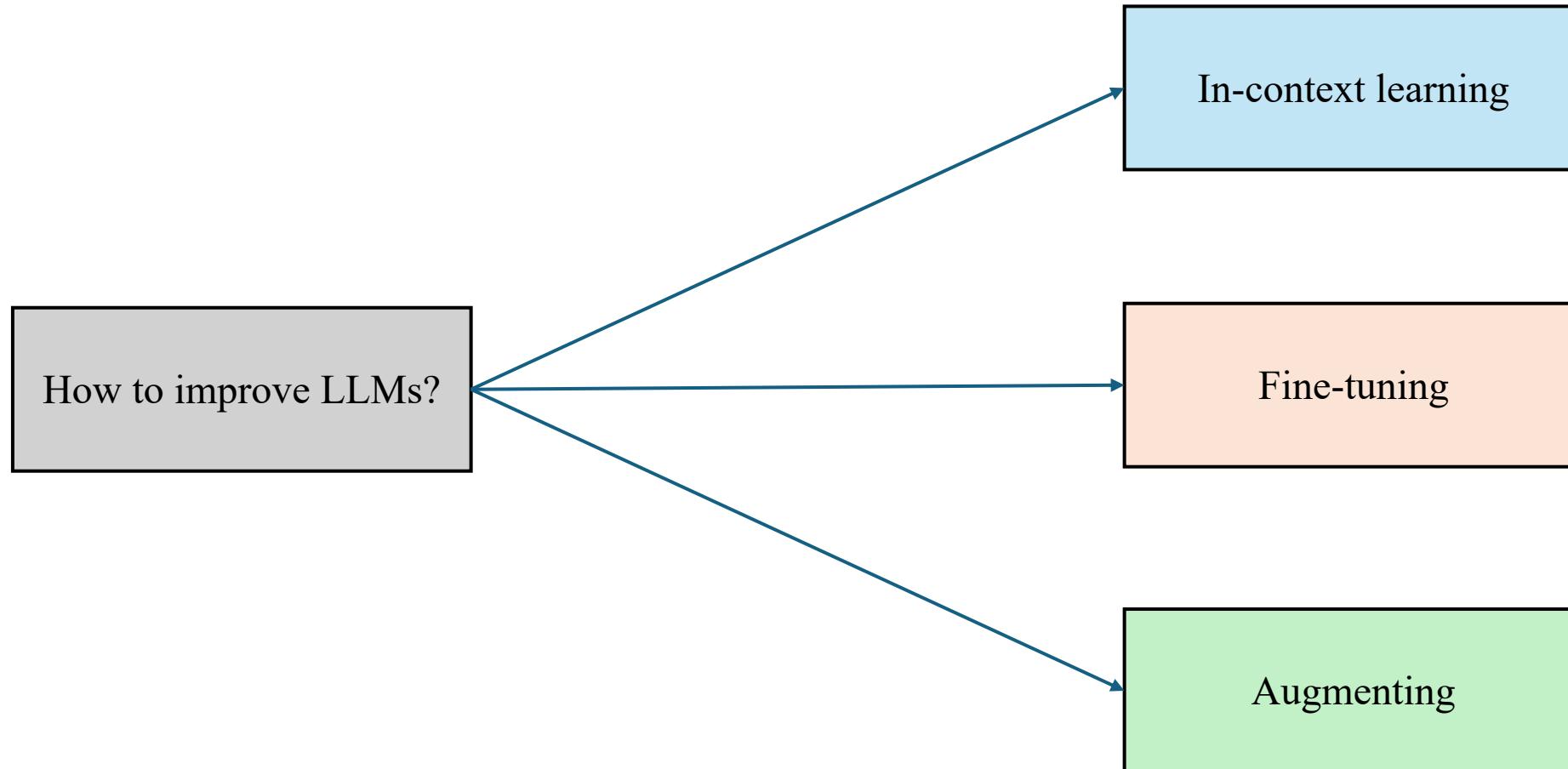
$241 - (-241) + 1$ is equivalent to $241 + 241 + 1$, which simplifies to $483 + 1$. So $241 - (-241) + 1$ is equal to 484.



Question: How can we improve LLMs on a specific task?

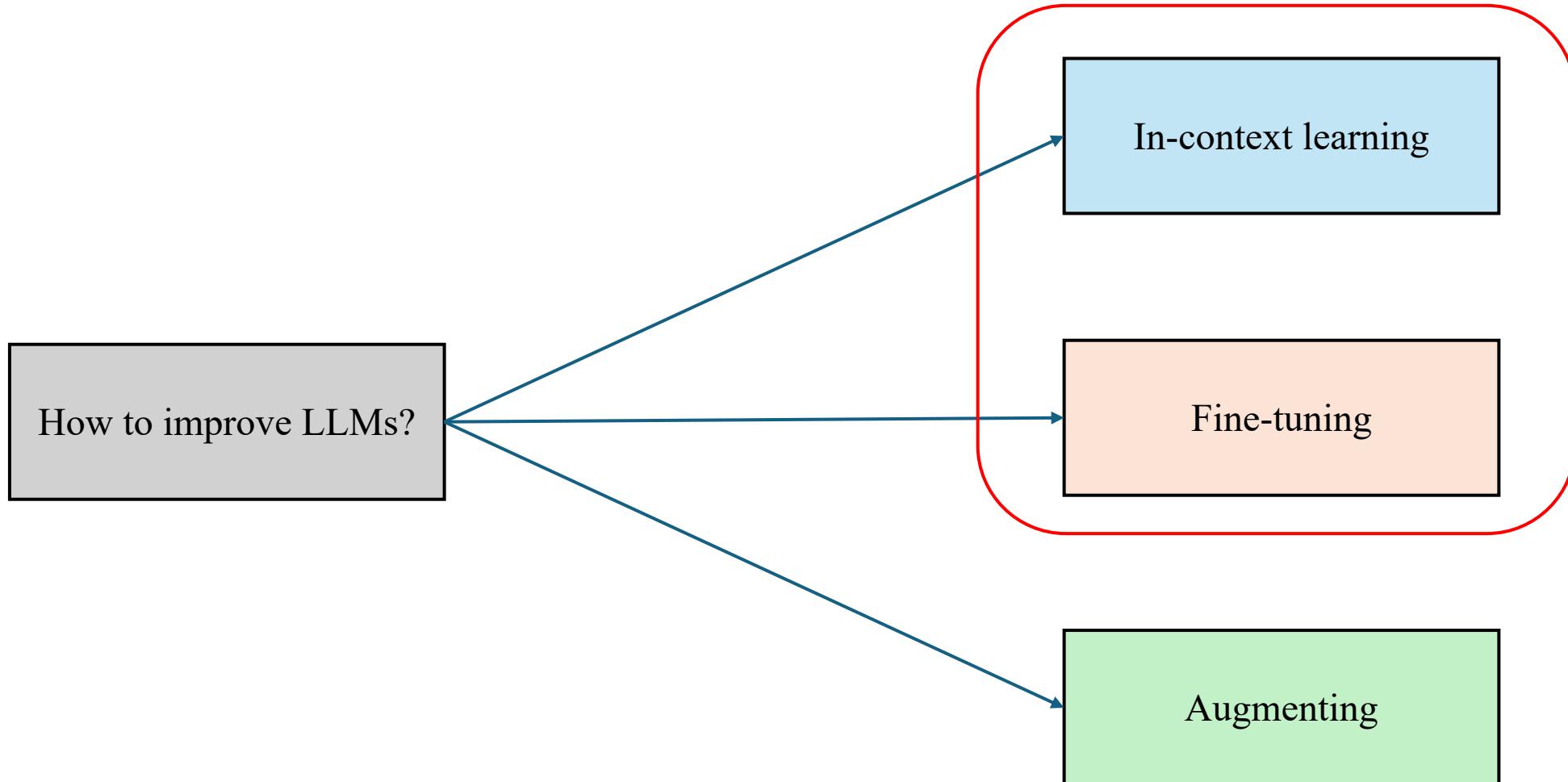
Introduction

❖ How to improve LLMs on specific tasks?



Introduction

❖ How to improve LLMs on specific tasks?

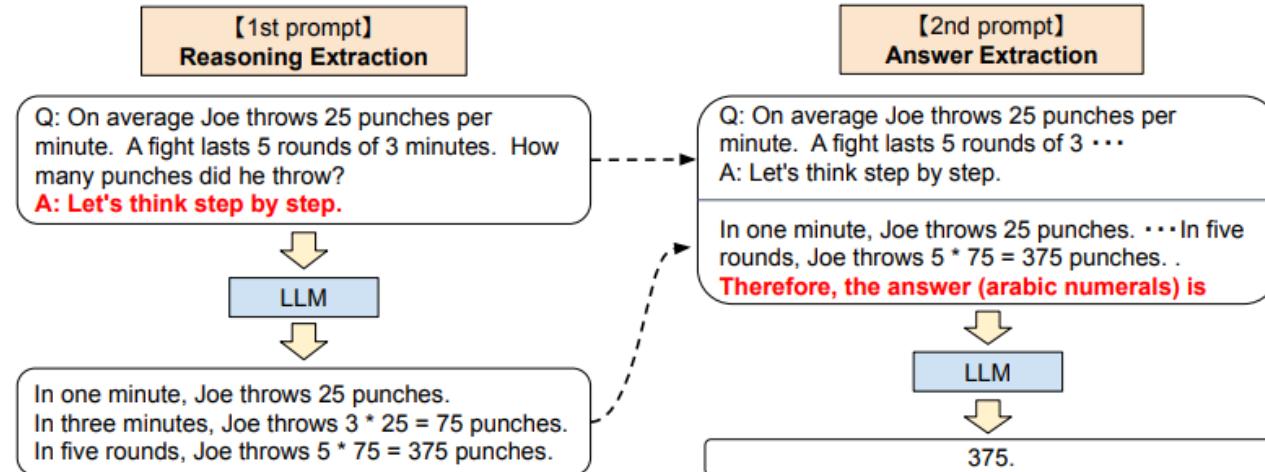




In-context Learning

In-context Learning

❖ Introduction



In-context learning (ICL) in LLMs is a technique where task demonstrations are integrated into the prompt in a natural language format. This approach allows pre-trained LLMs to address new tasks without fine-tuning the model.

In-context Learning

❖ Example

Input: 2025-04-20

Output: !04!20!2025

Input: 2024-05-29

Output: !05!29!2024

Input: 2018-07-03

Output: !07!03!2018

Input: 2025-04-23

Expected Output: !04!23!2025

In-context examples

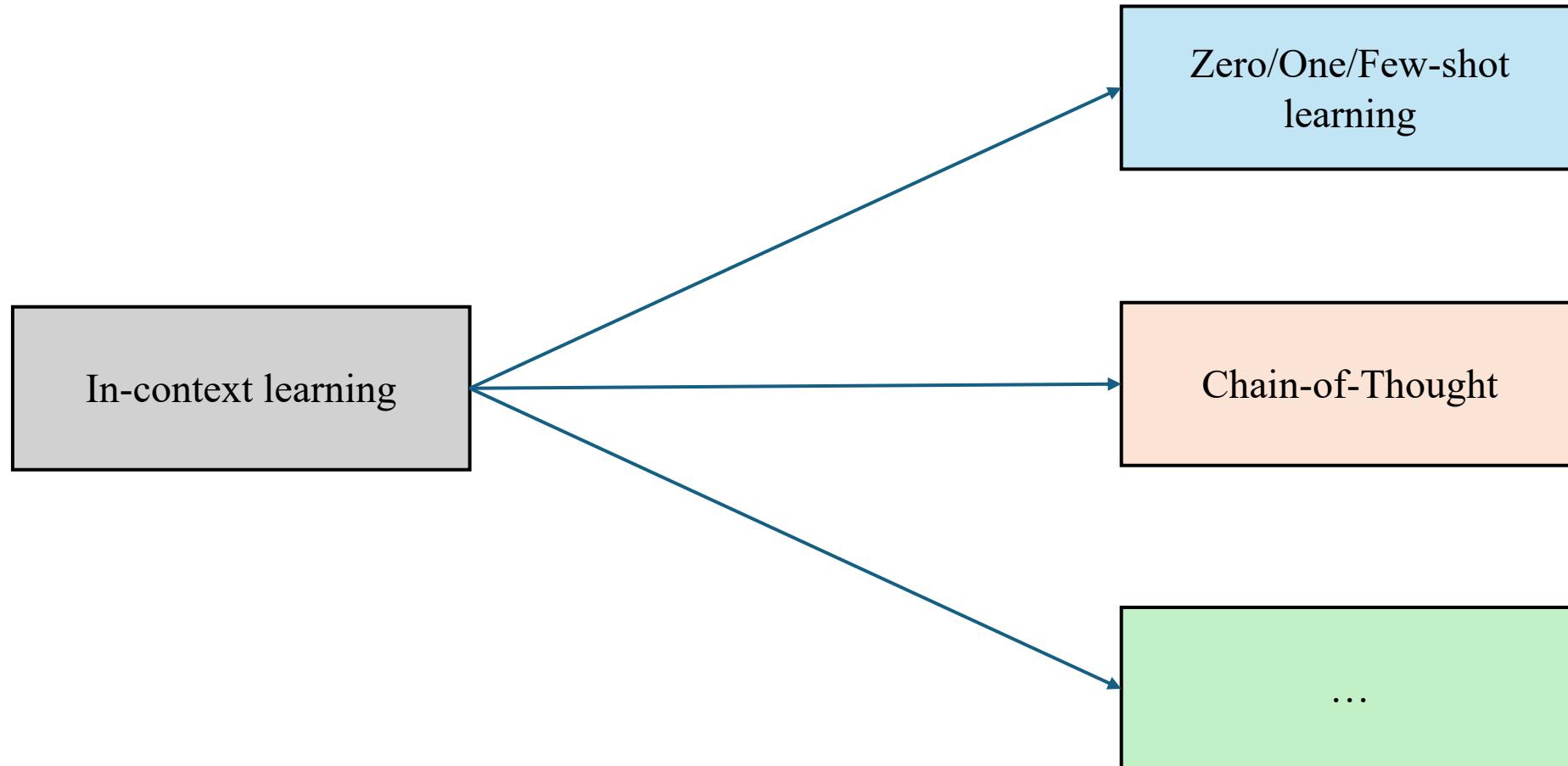
Test case

LLM Response

- ❖ Make LLMs adapt to a task using instruction and examples.

In-context Learning

❖ In-context learning type





In-context Learning

❖ Zero-shot learning

Prompt: Who won the 2014 World Cup?

GPT-4 Response: Germany won the 2014 FIFA World Cup. The tournament was held in Brazil, and Germany defeated Argentina 1–0 in the final, with the match-winning goal scored by Mario Götze during extra time

Zero-shot learning: Prompting LLMs to do a task without any examples.



In-context Learning

❖ One-shot learning

Prompt: Answer the question using the format shown in the context.

Who won the World Cup in **2014**?

Germany won the **2014** FIFA World Cup. They defeated Argentina in the final. It was a low scoring game.

Who won the world cup in **2018**?

GPT-4 Response: France won the **2018** FIFA World Cup. They defeated Croatia in the final. It was a higher scoring game compared to the **2014** final.

One-shot learning: Prompting LLMs to do a task with an example.



In-context Learning

❖ Few-shot learning

Prompt: Answer the question using the format shown in the context.

Who won the World Cup in **2014**?

Germany won the **2014** FIFA World Cup. They defeated Argentina in the final. It was a low scoring game.

Who won the World Cup in **2018**?

France won the **2018** FIFA World Cup. They defeated Croatia in the final. It was a high scoring game.

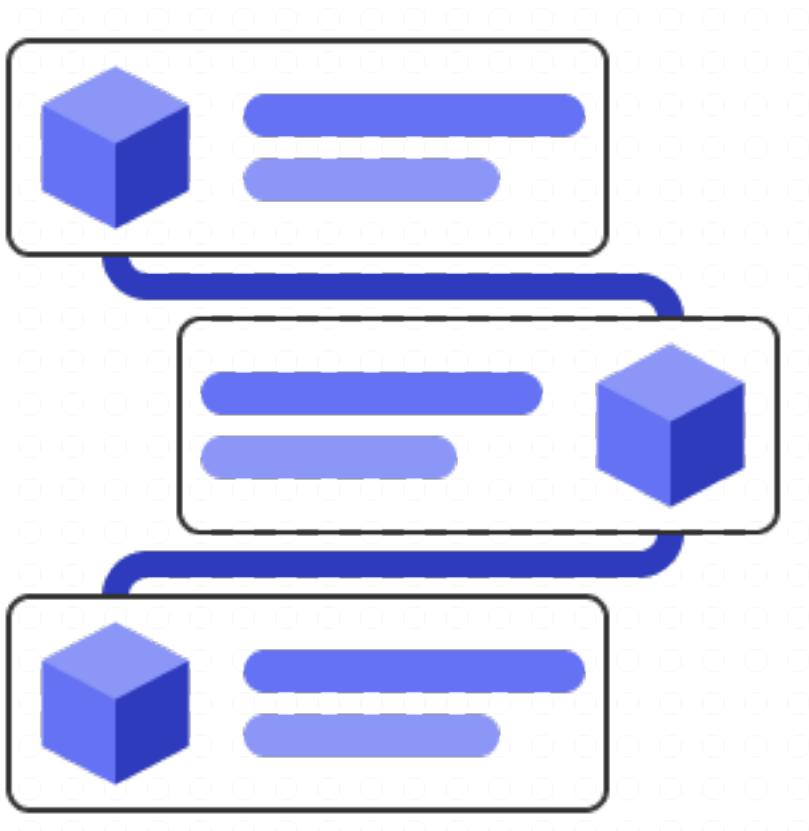
Who won the world cup in **2022**?

Few-shot learning: Prompting LLMs to do a task with more than one example.

GPT-4 Response: Argentina won the **2022** FIFA World Cup. They defeated France in the final. It was a high scoring game.

In-context Learning

❖ Chain-of-Thought Prompting



Chain-of-Thought (CoT) prompting enables complex reasoning capabilities through intermediate reasoning steps. You can combine it with few-shot prompting to get better results on more complex tasks that require reasoning before responding.

In-context Learning

❖ Zero-shot learning with CoT

(a) Few-shot

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: 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:

(Output) The answer is 8. **X**

(b) Few-shot-CoT

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: 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:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. **✓**

(c) Zero-shot

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: The answer (arabic numerals) is

(Output) 8 **X**

(d) Zero-shot-CoT (Ours)

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.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. **✓**

In-context Learning

❖ One-shot learning with CoT

Standard 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: 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

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?

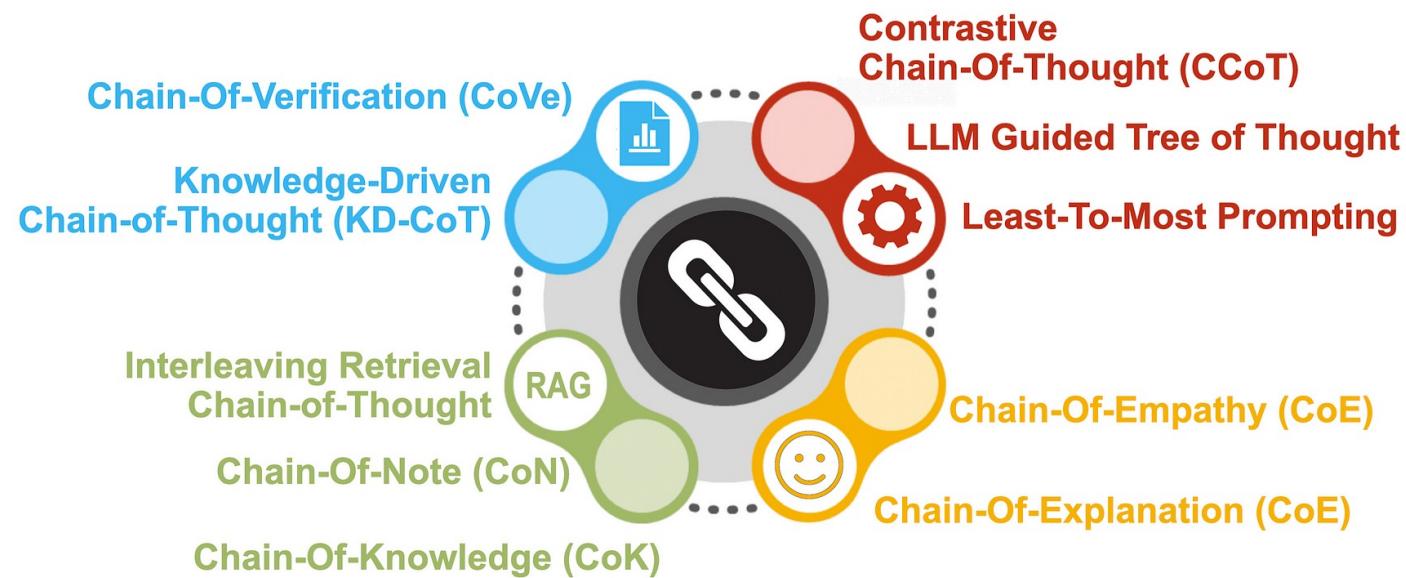
Model Output

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. ✓

In-context Learning

❖ CoT variations

Chain-of-Thought Prompting (CoT)

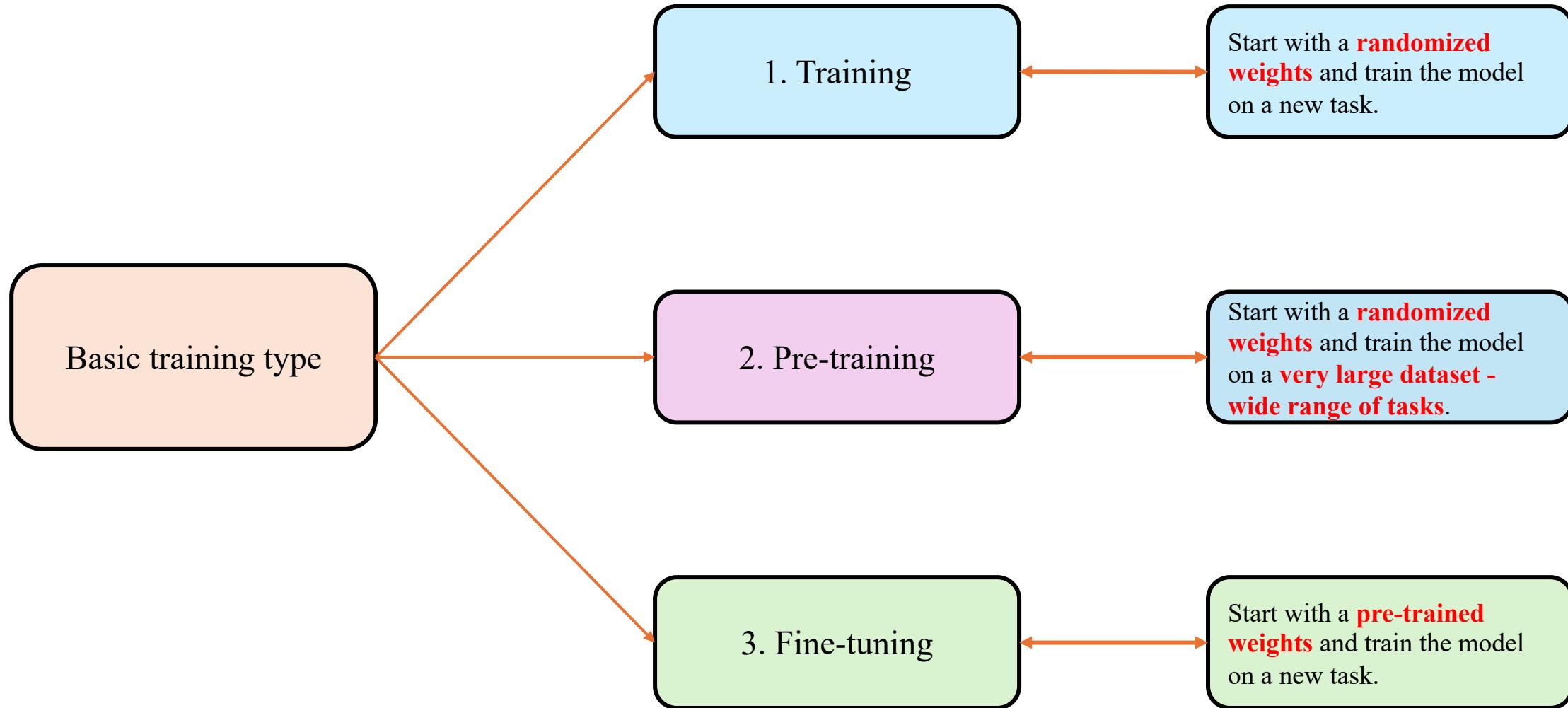




Supervised Fine-tuning

Supervised Fine-tuning

❖ Training model in Machine Learning



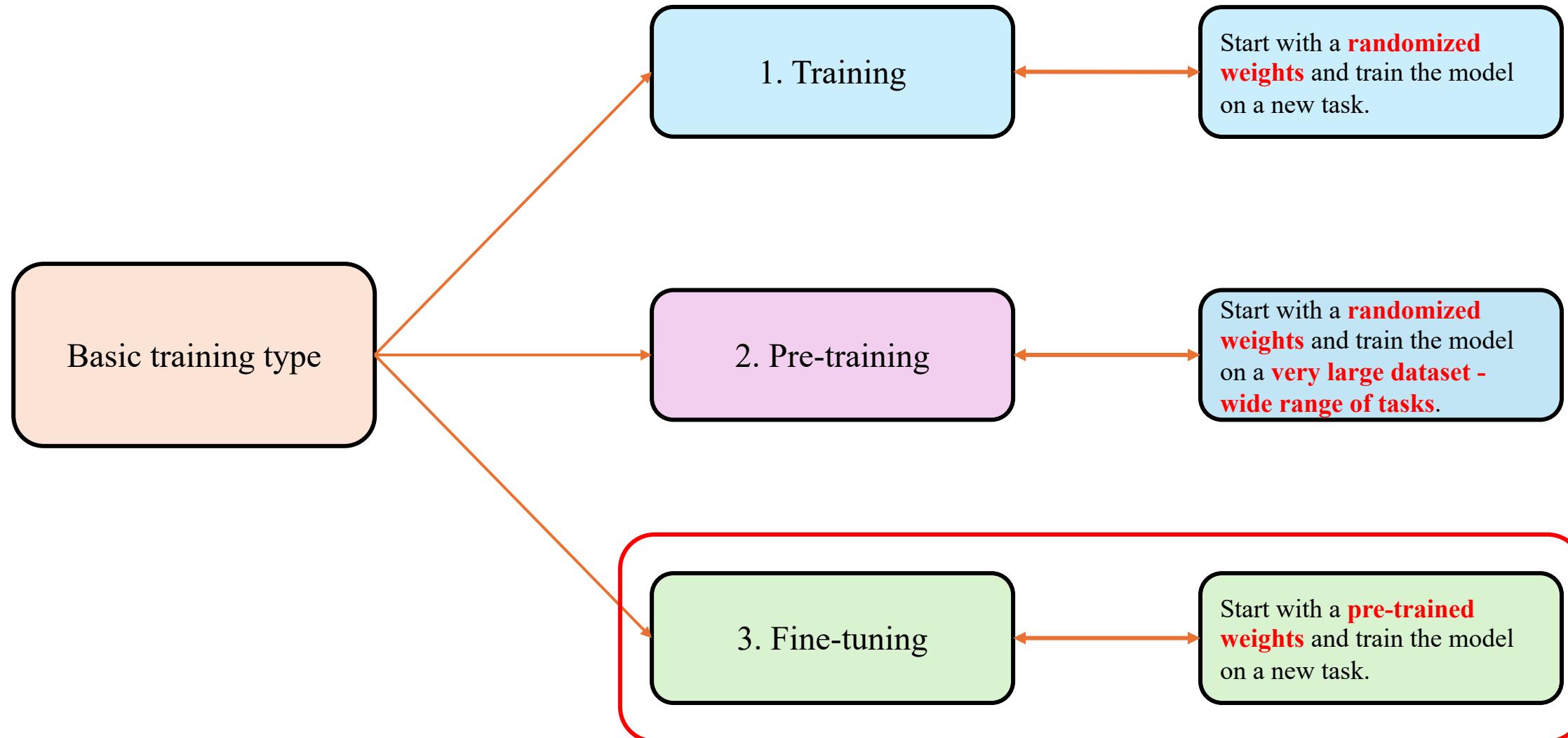
Supervised Fine-tuning

❖ Introduction

Supervised fine-tuning is the process of further training a pre-trained model on a labeled dataset to specialize it for a specific task. It leverages ground-truth input-output pairs to adjust the model's parameters for improved performance on the target application.

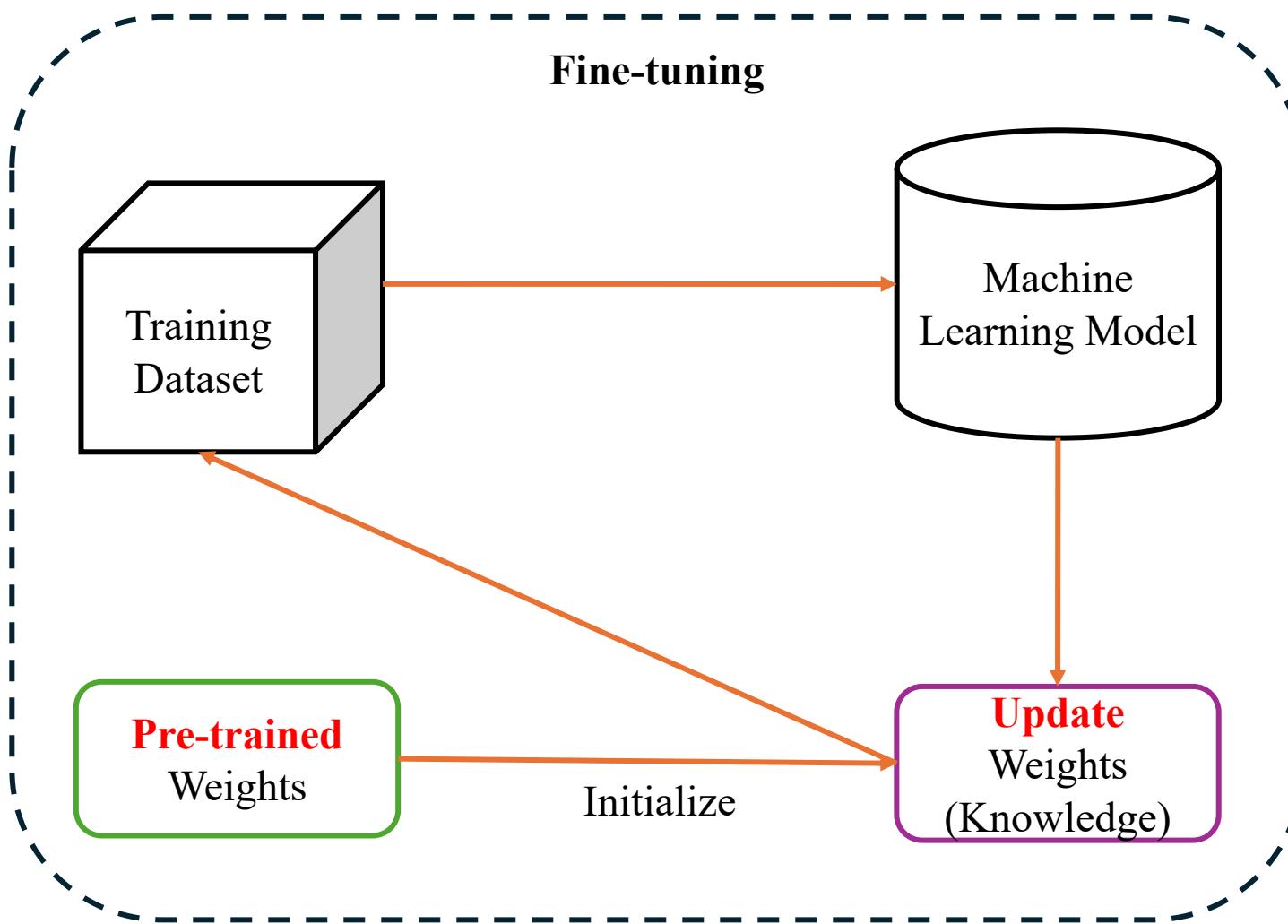
Supervised Fine-tuning

❖ Training model in Machine Learning



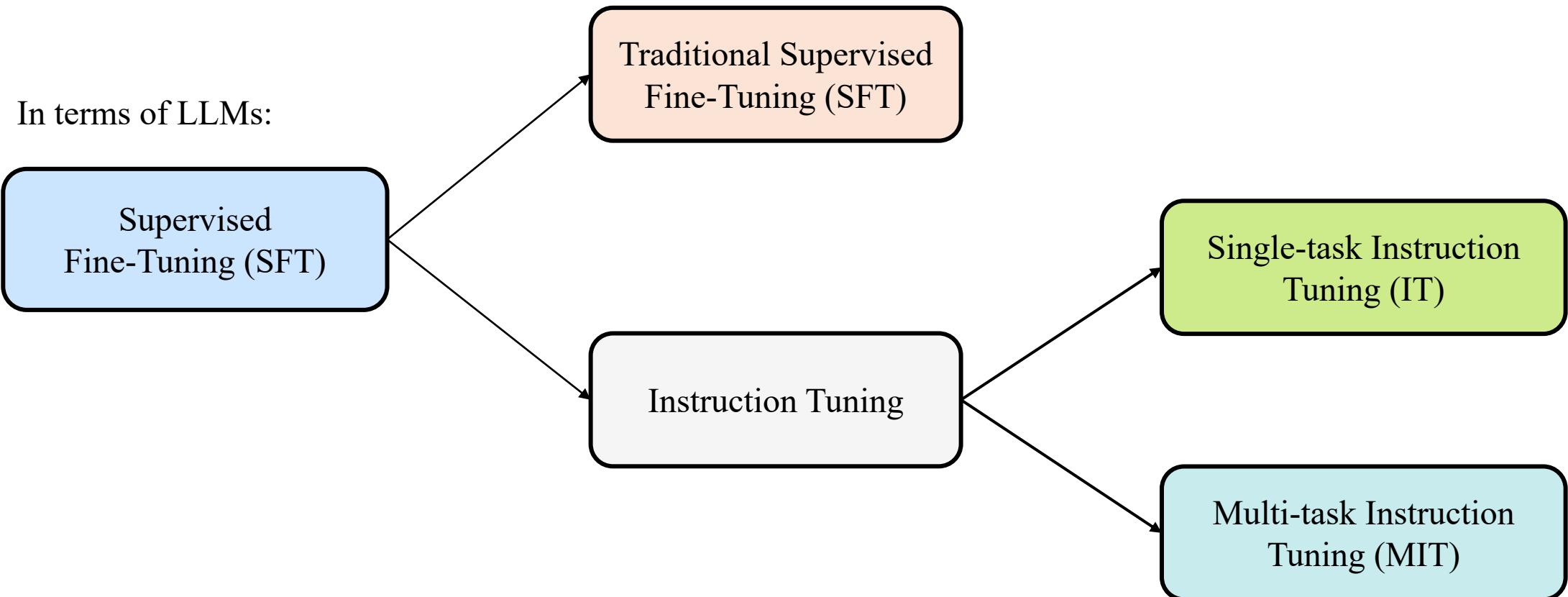
Supervised Fine-tuning

❖ Fine-tuning



Supervised Fine-tuning

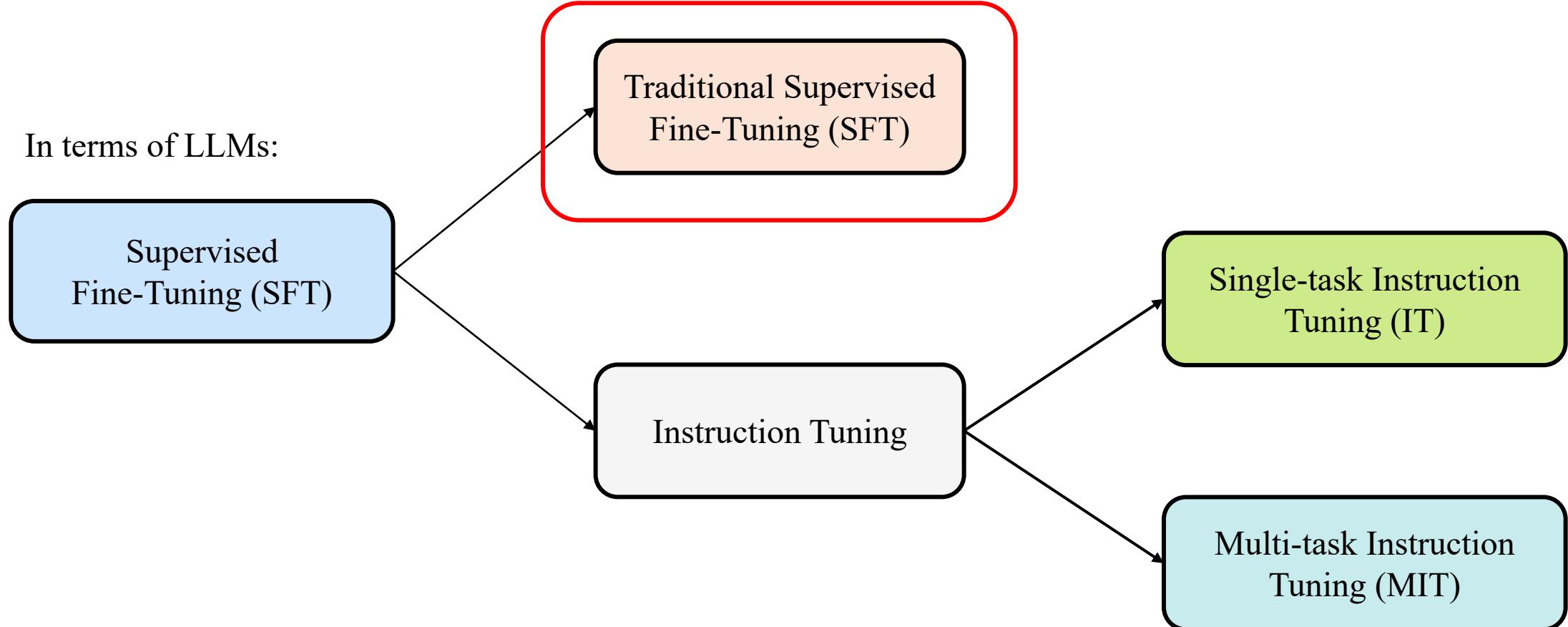
❖ LLM Fine-Tuning Strategies



Supervised Fine-tuning

❖ Supervised Fine-Tuning (SFT)

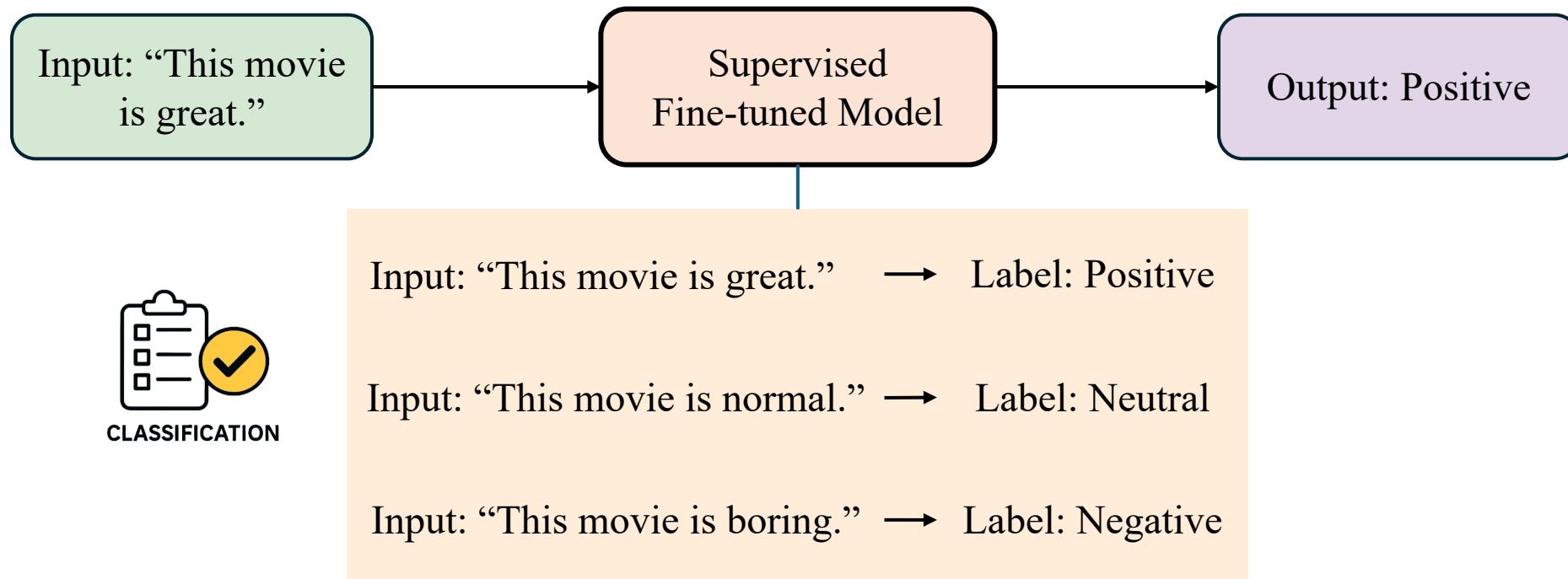
In terms of LLMs:



Supervised Fine-tuning

❖ Traditional Supervised Fine-Tuning (SFT)

Traditional Supervised Fine-Tuning (SFT) is the process of training a language model on task-specific input-output pairs, where each input is paired with a correct label or response. The goal is to optimize accuracy for known, structured tasks.

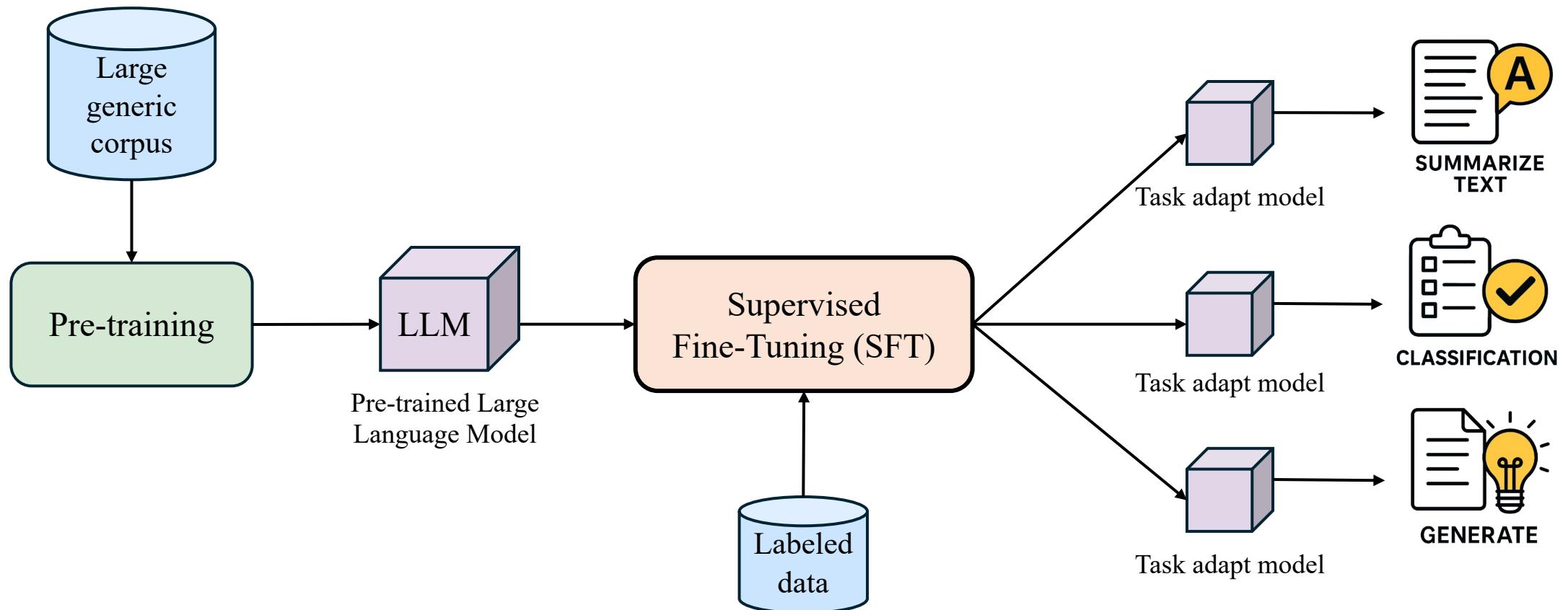


Training examples

Supervised Fine-tuning

❖ Supervised Fine-Tuning (SFT) in LLM

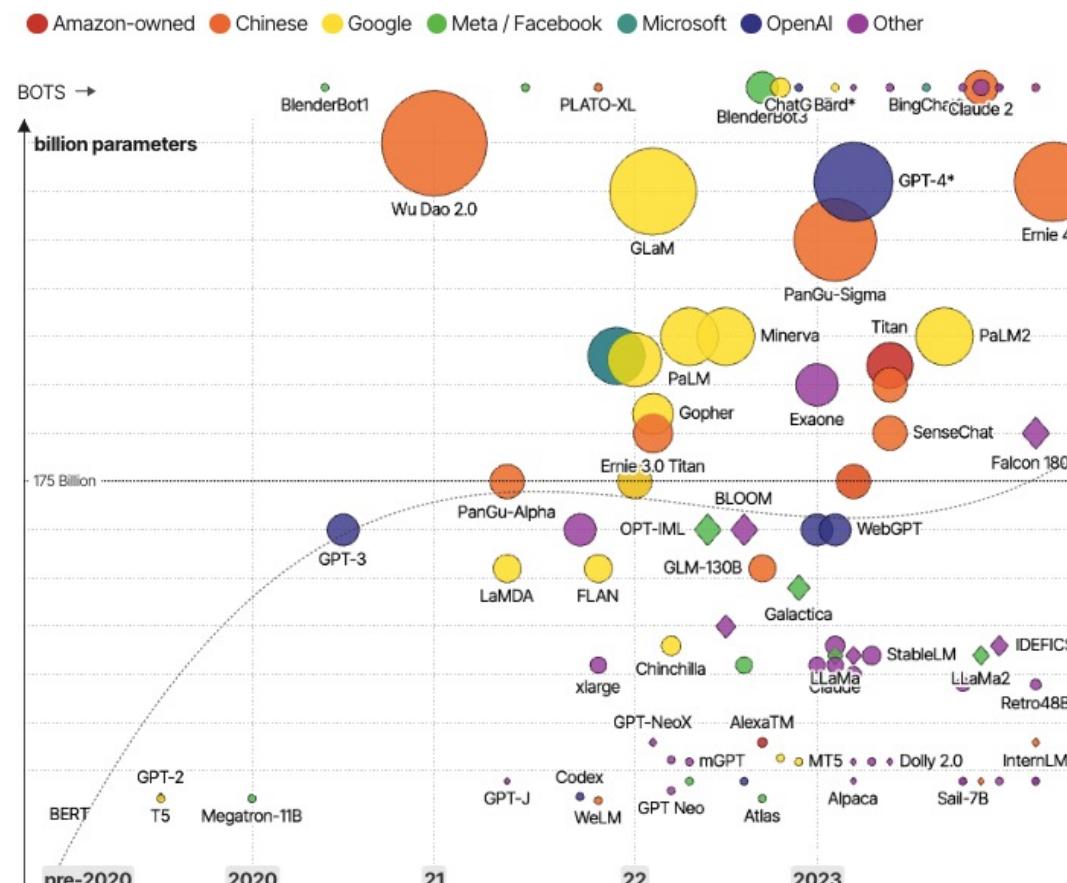
Supervised Fine-Tuning (SFT) is the process of training a language model on task-specific input-output pairs, where each input is paired with a correct label or response. The goal is to optimize accuracy for known, structured tasks.



Supervised Fine-tuning

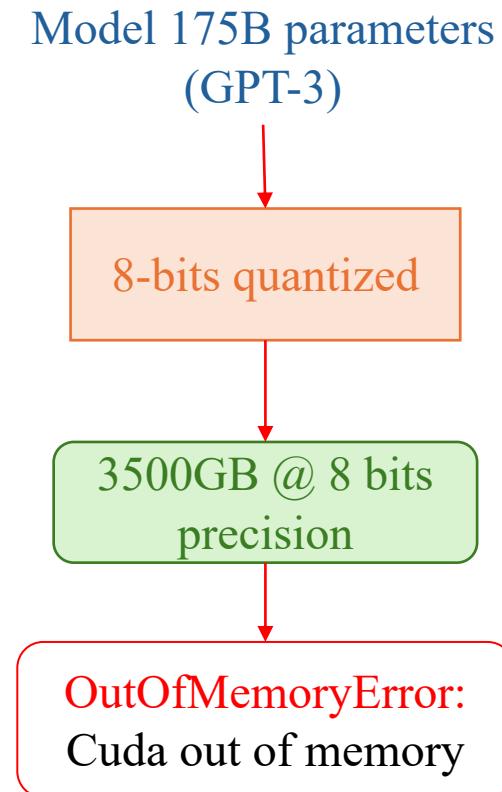
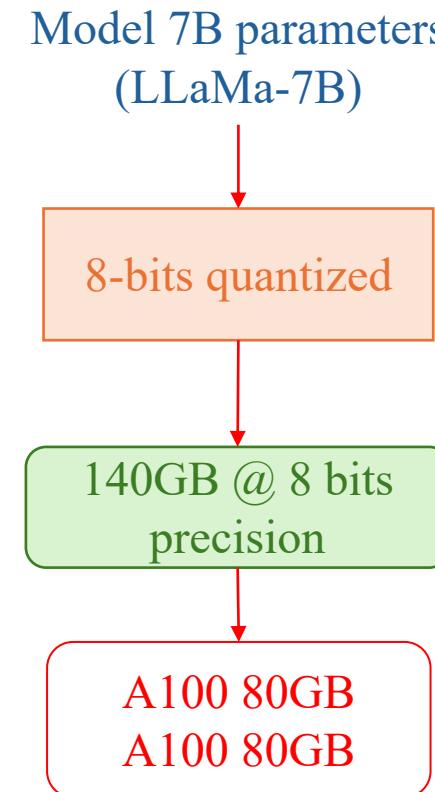
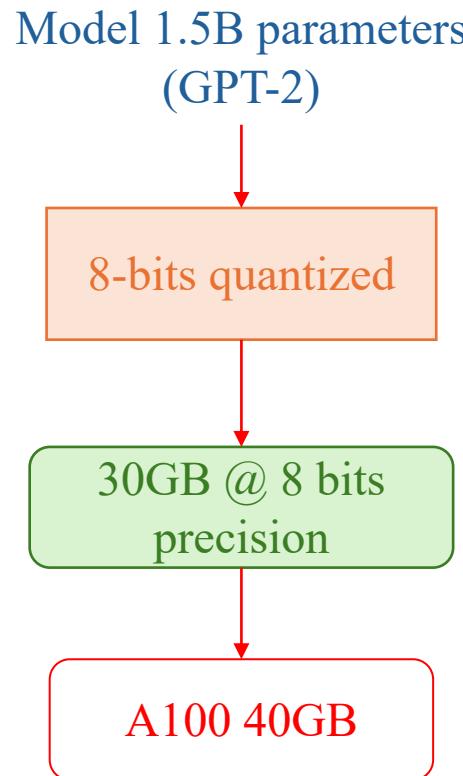
❖ Problem Description

The Rise and Rise of A.I. Large Language Models (LLMs) & their associated bots like ChatGPT



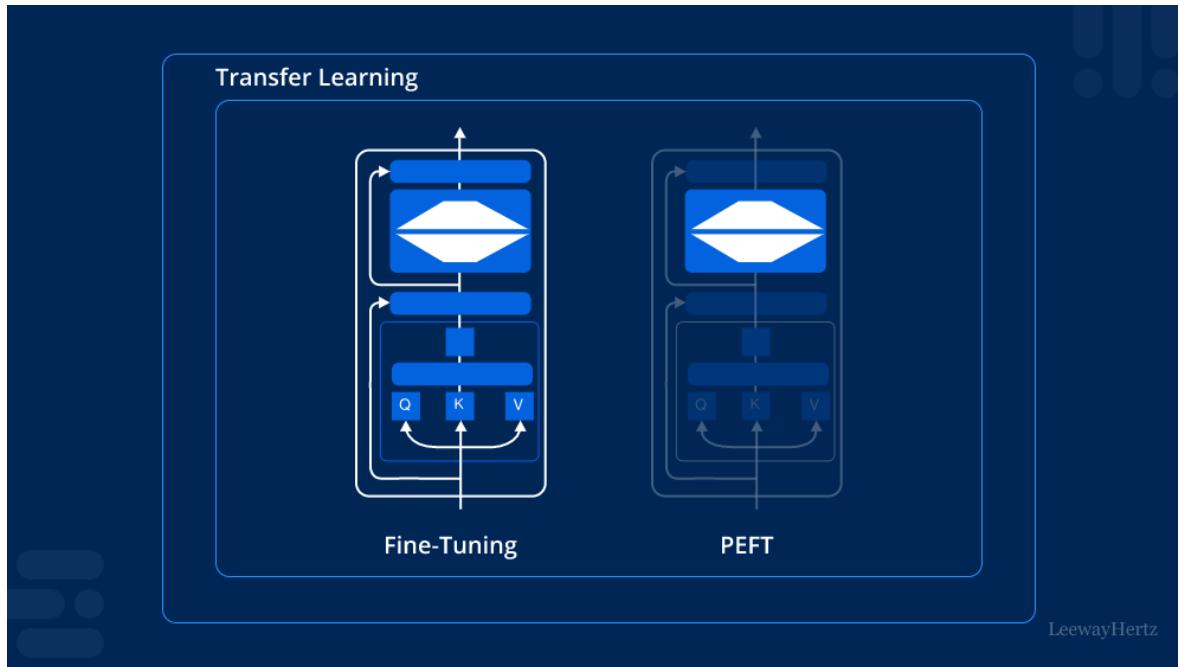
Supervised Fine-tuning

❖ Problem Description



Supervised Fine-tuning

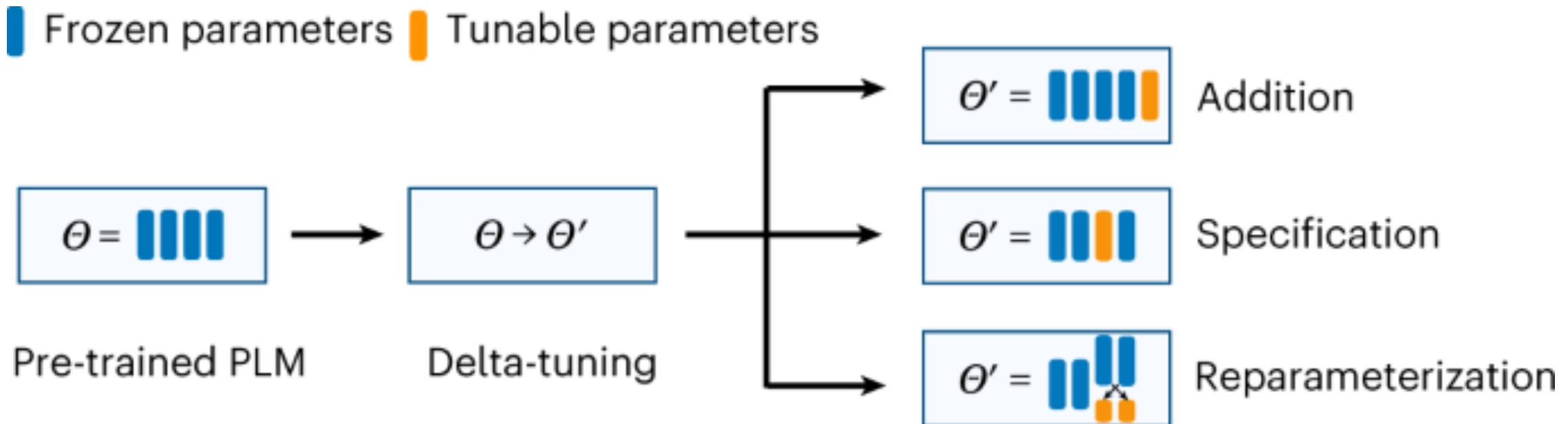
❖ PEFT Type



Parameter-efficient Fine-tuning (PEFT) is a technique used to improve the performance of pre-trained language models on specific downstream tasks. It involves reusing the pre-trained model's parameters and fine-tuning them on a smaller dataset, which saves computational resources and time compared to training the entire model from scratch.

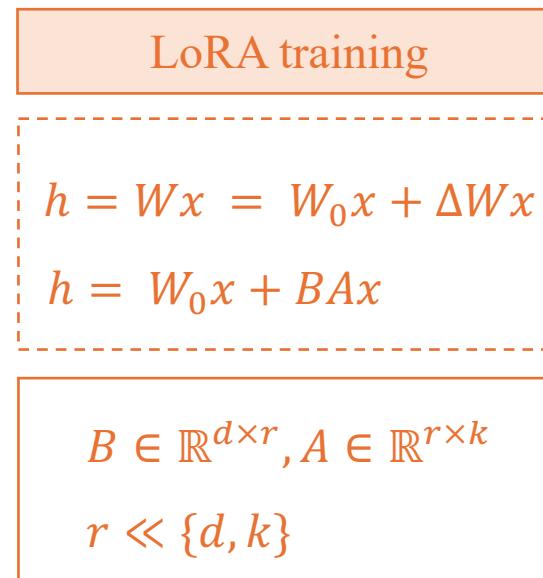
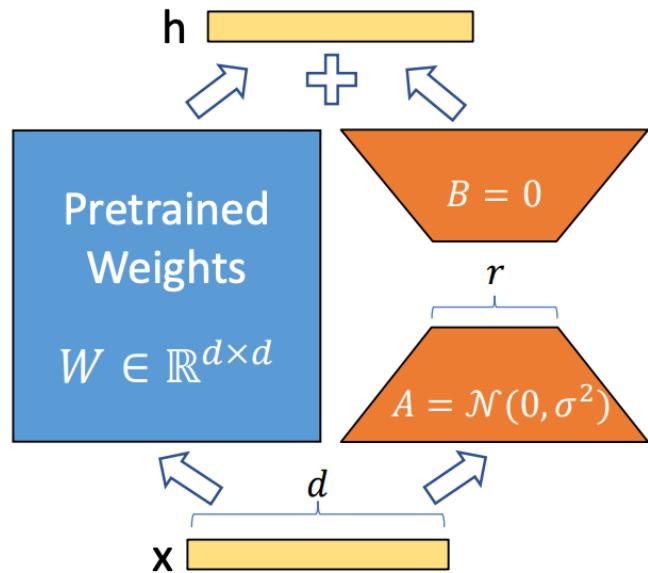
Supervised Fine-tuning

❖ PEFT Type

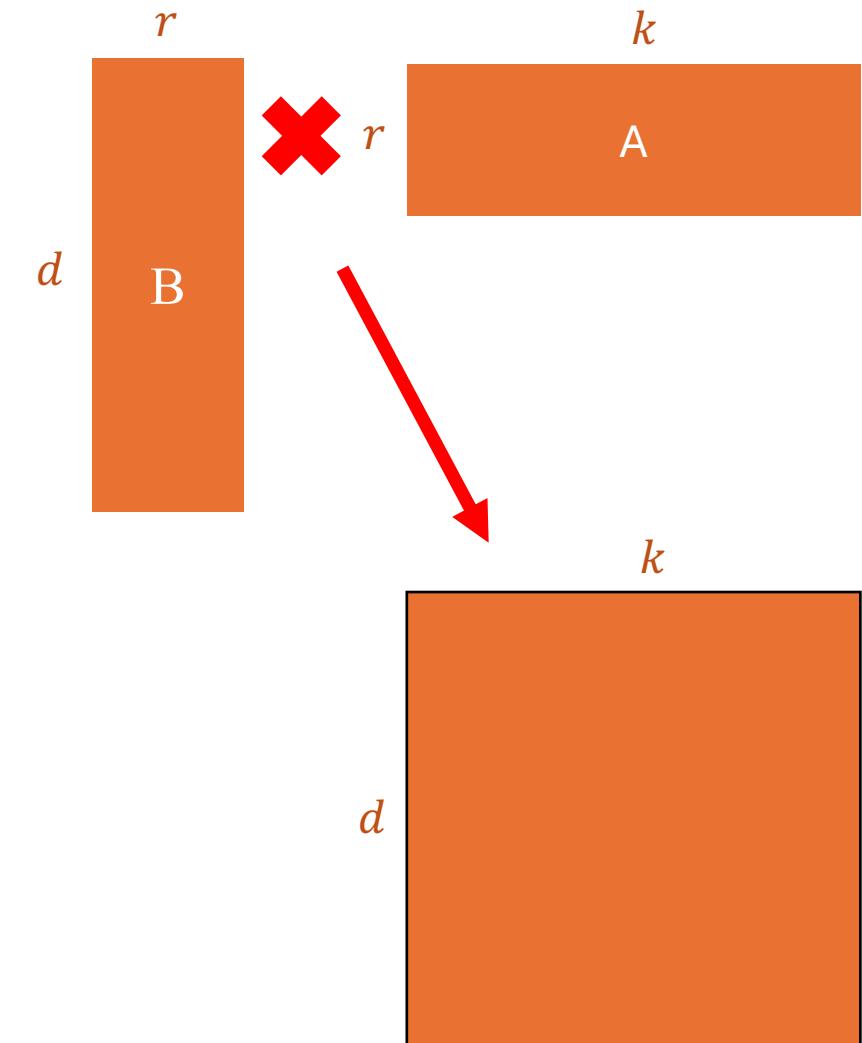


Supervised Fine-tuning

❖ LoRA: Low Rank Adaptation

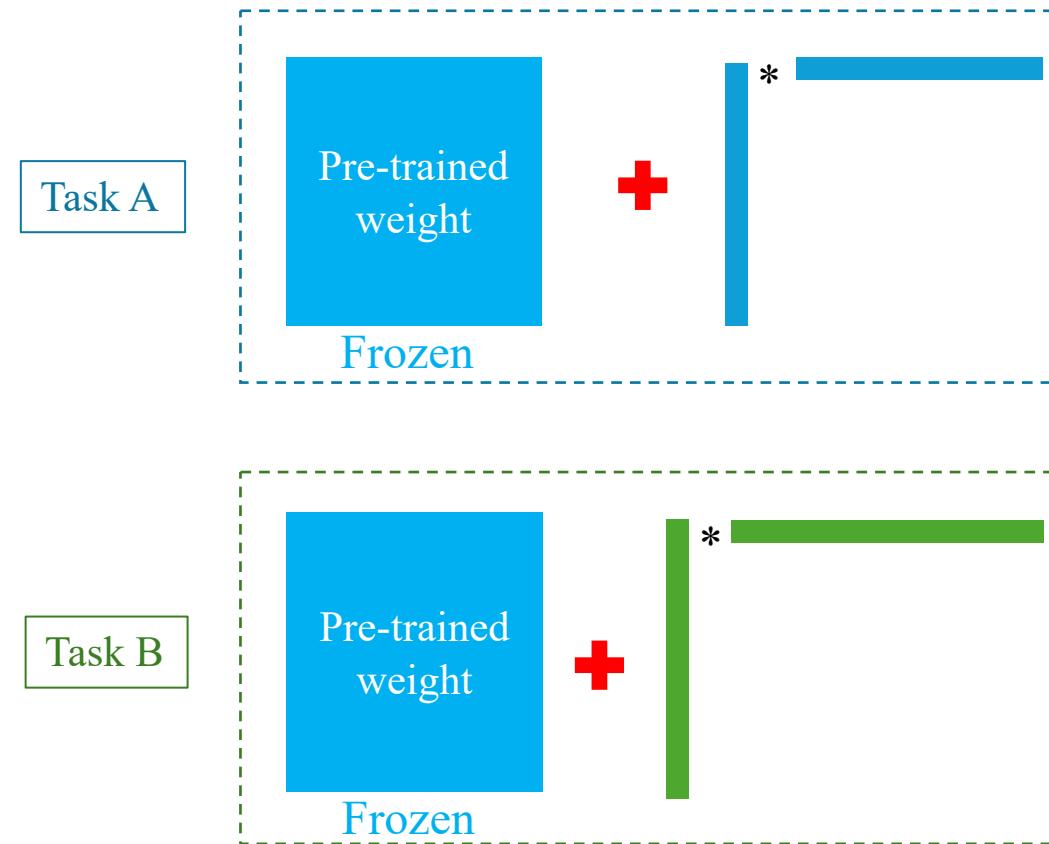
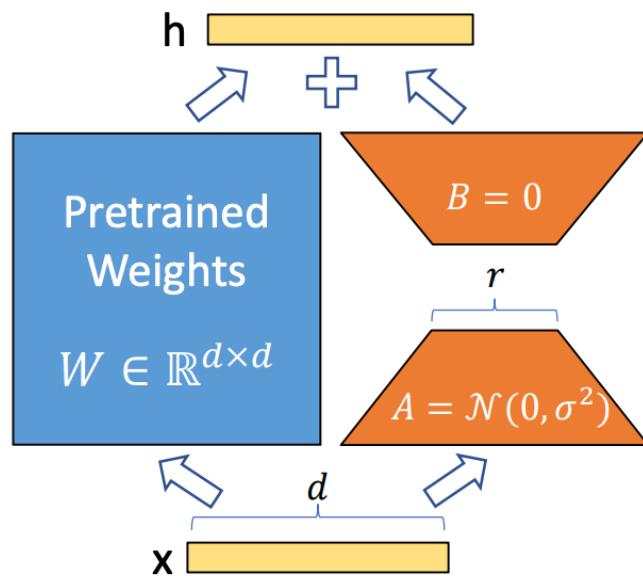


* \hat{L}_{LoRA} is the number of linear layers applied to LoRA



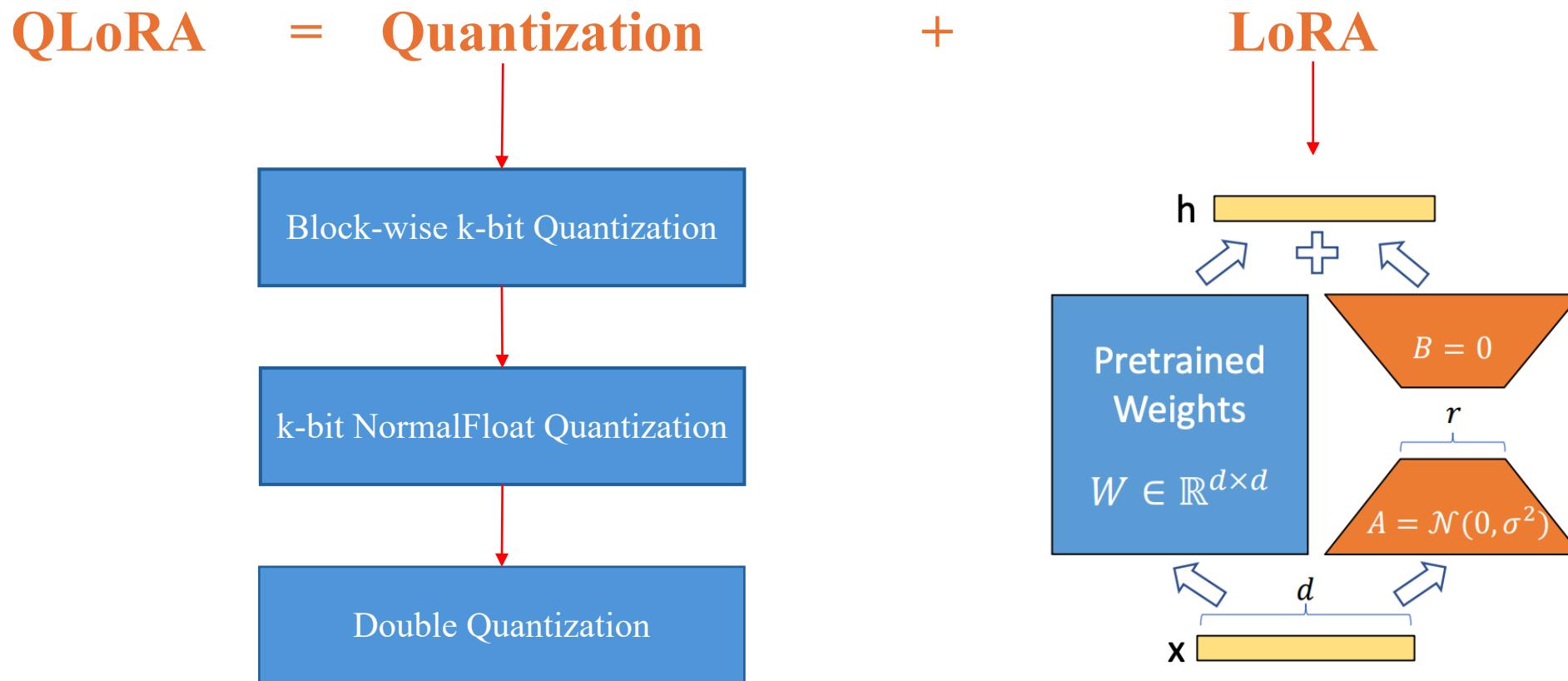
Supervised Fine-tuning

❖ LoRA: Low Rank Adaptation



Supervised Fine-tuning

❖ QLoRA



Supervised Fine-tuning

❖ Performance of LoRA

Epoch [1/10], Average Loss: 1.3470, GPU used: 2.06 G
Epoch [2/10], Average Loss: 1.1362, GPU used: 2.06 G
Epoch [3/10], Average Loss: 1.1051, GPU used: 2.06 G
Epoch [4/10], Average Loss: 1.0829, GPU used: 2.06 G
Epoch [5/10], Average Loss: 1.0814, GPU used: 2.06 G
Epoch [6/10], Average Loss: 1.0621, GPU used: 2.06 G
Epoch [7/10], Average Loss: 1.0474, GPU used: 2.06 G
Epoch [8/10], Average Loss: 1.0420, GPU used: 2.06 G
Epoch [9/10], Average Loss: 1.0216, GPU used: 2.06 G
Epoch [10/10], Average Loss: 1.0221, GPU used: 2.06 G
Finished Training
Training time: 204.78 s

Without LoRA.

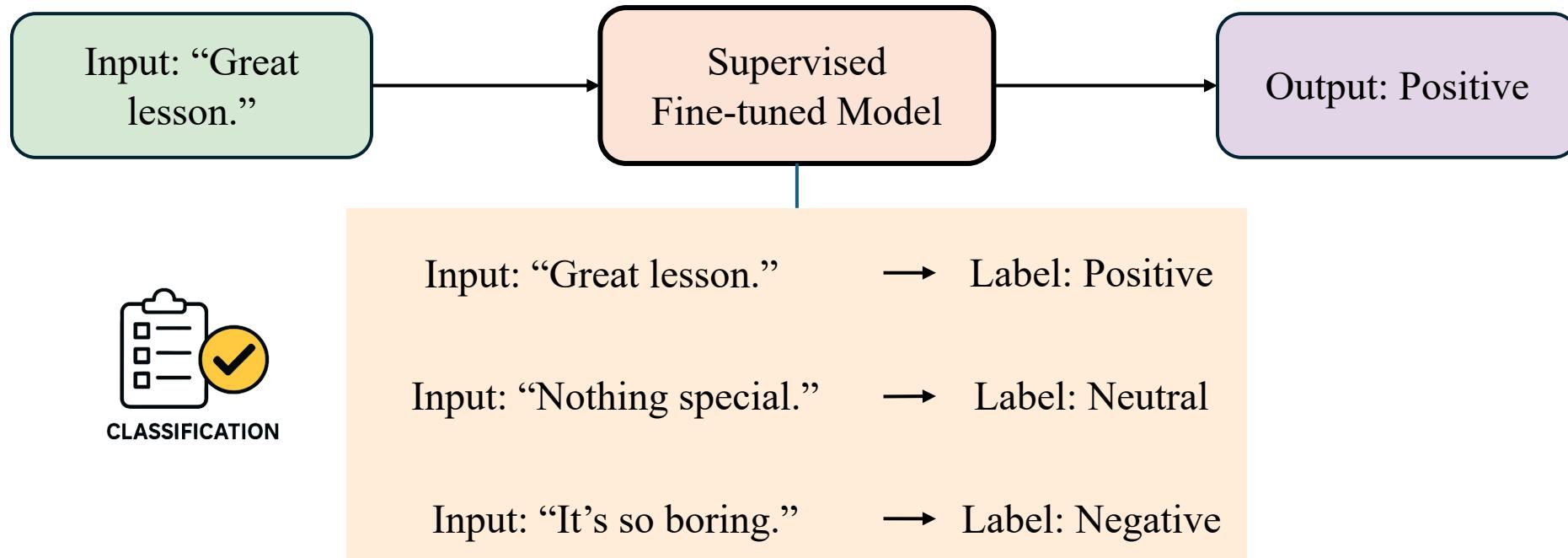
Epoch [1/10], Average Loss: 1.5478, GPU used: 0.59 G
Epoch [2/10], Average Loss: 1.2768, GPU used: 0.59 G
Epoch [3/10], Average Loss: 1.2090, GPU used: 0.59 G
Epoch [4/10], Average Loss: 1.1460, GPU used: 0.59 G
Epoch [5/10], Average Loss: 1.1072, GPU used: 0.59 G
Epoch [6/10], Average Loss: 1.0763, GPU used: 0.59 G
Epoch [7/10], Average Loss: 1.0581, GPU used: 0.59 G
Epoch [8/10], Average Loss: 1.0360, GPU used: 0.59 G
Epoch [9/10], Average Loss: 1.0221, GPU used: 0.59 G
Epoch [10/10], Average Loss: 0.9981, GPU used: 0.59 G
Finished Training
Training time: 128.50 s

With LoRA.

Supervised Fine-tuning

❖ Problem Description: SFT

Description: Build a Vietnamese sentiment classification system to analyze student feedbacks and classify them into negative, neutral, or positive categories.



Supervised Fine-tuning

❖ Problem Description: SFT

Description: Build a Vietnamese sentiment classification system to analyze student feedbacks and classify them into negative, neutral, or positive categories.



Supervised Fine-tuning

❖ Step 1: Install and import necessary libraries

```
!pip install -q -U bitsandbytes
!pip install -q -U datasets
!pip install -q -U git+https://github.com/huggingface/transformers.git
!pip install -q -U git+https://github.com/huggingface/peft.git
!pip install -q -U git+https://github.com/huggingface/accelerate.git
!pip install -q -U loralib
!pip install -q -U einops
```

```
1 import os
2 import torch
3 import numpy as np
4 from sklearn.metrics import accuracy_score
5 from datasets import load_dataset
6
7 from transformers import (
8     AutoModelForSequenceClassification,
9     AutoTokenizer,
10    TrainingArguments,
11    Trainer,
12    TrainerCallback,
13    DataCollatorWithPadding,
14    DataCollatorForLanguageModeling,
15    BitsAndBytesConfig,
16    set_seed
17 )
18
19 from peft import (
20     LoraConfig,
21     get_peft_model,
22     prepare_model_for_kbit_training,
23     PeftModel,
24     PeftConfig
25 )
```

Supervised Fine-tuning

❖ Step 2: Load & Preprocess Dataset

```
1 dataset = load_dataset("uit-nlp/vietnamese_students_feedback", trust_remote_code=True)

/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
README.md: 100% [██████████] 4.79k/4.79k [00:00<00:00, 196kB/s]
vietnamese_students_feedback.py: 100% [██████████] 4.97k/4.97k [00:00<00:00, 424kB/s]
Downloading data: 100% [██████████] 898k/898k [00:00<00:00, 6.06MB/s]
Downloading data: 100% [██████████] 22.9k/22.9k [00:00<00:00, 613kB/s]
Downloading data: 100% [██████████] 22.9k/22.9k [00:00<00:00, 1.04MB/s]
Downloading data: 100% [██████████] 119k/119k [00:00<00:00, 2.68MB/s]
Downloading data: 100% [██████████] 3.17k/3.17k [00:00<00:00, 82.0kB/s]
Downloading data: 100% [██████████] 3.17k/3.17k [00:00<00:00, 68.8kB/s]
Downloading data: 100% [██████████] 248k/248k [00:00<00:00, 4.30MB/s]
Downloading data: 100% [██████████] 6.33k/6.33k [00:00<00:00, 718kB/s]
Downloading data: 100% [██████████] 6.33k/6.33k [00:00<00:00, 655kB/s]
Generating train split: [████] 11426/0 [00:00<00:00, 23118.52 examples/s]
Generating validation split: [████] 1583/0 [00:00<00:00, 27166.87 examples/s]
Generating test split: [████] 3166/0 [00:00<00:00, 33179.00 examples/s]
```

```
1 print("Dataset structure:", dataset)

Dataset structure: DatasetDict({
  train: Dataset({
    features: ['sentence', 'sentiment', 'topic'],
    num_rows: 11426
  })
  validation: Dataset({
    features: ['sentence', 'sentiment', 'topic'],
    num_rows: 1583
  })
  test: Dataset({
    features: ['sentence', 'sentiment', 'topic'],
    num_rows: 3166
  })
})
```

Supervised Fine-tuning

❖ Step 2: Load & Preprocess Dataset

Student feedbacks
(sentence ->text)

sentence
string · lengths



sentiment -> label:
(0:negative,
1: neutral,
2: positive)

Topic is not really
necessary -> Remove

	sentiment	topic
	3 classes	4 classes
slide giáo trình đầy đủ .	2 positive	1 training_program
nhiệt tình giảng dạy , gần gũi với sinh viên .	2 positive	0 lecturer
đi học đầy đủ full điểm chuyên cần .	0 negative	1 training_program
chưa áp dụng công nghệ thông tin và các thiết bị hỗ trợ cho việc giảng dạy .	0 negative	0 lecturer
thầy giảng bài hay , có nhiều bài tập ví dụ ngay trên lớp .	2 positive	0 lecturer
giảng viên đảm bảo thời gian lên lớp , tích cực trả lời câu hỏi của sinh viên , thường xuyên đặt câu hỏi cho sinh viên .	2 positive	0 lecturer
em sẽ nợ môn này , nhưng em sẽ học lại ở các học kỳ kế tiếp .	1 neutral	3 others
thời lượng học quá dài , không đảm bảo tiếp thu hiệu quả .	0 negative	1 training_program

< Previous 1 2 3 ... 115 Next >

[uit-nlp/vietnamese_students_feedback](#) Dataset.

```
1 def preprocess(example):
2     return {
3         "text": str(example["sentence"]),
4         "label": int(example["sentiment"])
5     }
6
7 dataset = dataset.map(
8     preprocess,
9     remove_columns=["sentence", "sentiment", "topic"]
10 )
```

Map: 100%	11426/11426 [00:00<00:00, 20163.85 examples/s]
Map: 100%	1583/1583 [00:00<00:00, 16040.66 examples/s]
Map: 100%	3166/3166 [00:00<00:00, 16784.73 examples/s]

1 dataset

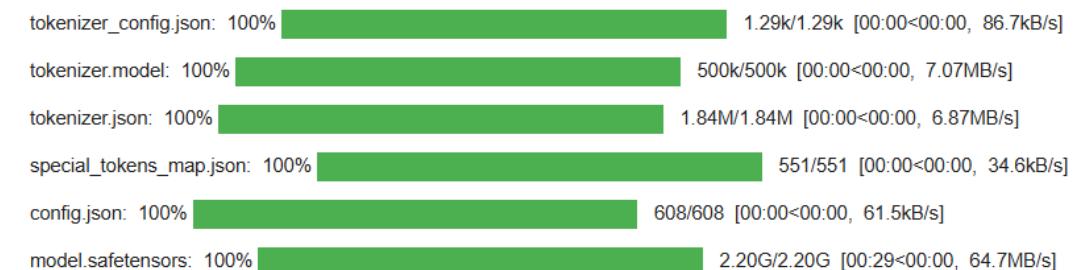
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 11426
    })
    validation: Dataset({
        features: ['text', 'label'],
        num_rows: 1583
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 3166
    })
})
```

Supervised Fine-tuning

❖ Step 3: Load Pre-trained Model

```
1 model_id = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"
2 tokenizer = AutoTokenizer.from_pretrained(model_id)
3 tokenizer.pad_token = tokenizer.eos_token
4
5 bnb_config = BitsAndBytesConfig(
6     load_in_4bit=True,
7     bnb_4bit_use_double_quant=True,
8     bnb_4bit_quant_type="nf4",
9     bnb_4bit_compute_dtype=torch.float16
10)
11
12 model = AutoModelForSequenceClassification.from_pretrained(
13     model_id,
14     num_labels=3,    3 sentiment labels
15     quantization_config=bnb_config,
16     id2label={0: "tiêu_cực", 1: "trung_lập", 2: "tích_cực"},
17     label2id={"tiêu_cực": 0, "trung_lập": 1, "tích_cực": 2},
18     pad_token_id=tokenizer.pad_token_id
19 )
```

Configure 4-bit quantization.



maps numeric labels to human-readable strings and vice versa.

Supervised Fine-tuning

❖ Step 3: Load Pre-trained Model

```
21 model = prepare_model_for_kbit_training(model)
22 peft_config = LoraConfig(
23     r=8,
24     lora_alpha=16,
25     target_modules=["q_proj", "v_proj"],
26     lora_dropout=0.05,
27     bias="none",
28     task_type="SEQ_CLS",
29     inference_mode=False
30 )
31 model = get_peft_model(model, peft_config)
```

Adapts the quantized model to make it compatible with PEFT (LoRA).

```
1 model.print_trainable_parameters()
```

```
trainable params: 1,132,544 || all params: 1,035,651,072 || trainable%: 0.1094
```

Supervised Fine-tuning

❖ Step 4: Tokenization

```
1 def tokenize_function(examples):
2     tokenized = tokenizer(
3         examples["text"],
4         truncation=True,
5         max_length=256,
6         padding="max_length",
7         add_special_tokens=True
8     )
9     return {
10        "input_ids": tokenized["input_ids"],
11        "attention_mask": tokenized["attention_mask"],
12        "labels": examples["label"]
13    }
14
15 tokenized_dataset = dataset.map(
16    tokenize_function,
17    batched=True,
18    remove_columns=["text", "label"]
19 )
```

```
1 print("\nDataset sau khi xử lý:")
2 print(tokenized_dataset)

Dataset sau khi xử lý:
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'attention_mask', 'labels'],
        num_rows: 11426
    })
    validation: Dataset({
        features: ['input_ids', 'attention_mask', 'labels'],
        num_rows: 1583
    })
    test: Dataset({
        features: ['input_ids', 'attention_mask', 'labels'],
        num_rows: 3166
    })
})
```

Map: 100%  11426/11426 [00:02<00:00, 3909.12 examples/s]

Map: 100%  1583/1583 [00:00<00:00, 4895.99 examples/s]

Map: 100%  3166/3166 [00:00<00:00, 6536.68 examples/s]

Supervised Fine-tuning

❖ Step 4: Tokenization

```
1 def test_output_before_training(model, tokenizer, samples=None):
2     if samples is None:
3         samples = [
4             "Giảng viên giảng bài dễ hiểu và nhiệt tình.",
5             "Em cảm thấy rất áp lực với lịch học dày đặc.",
6             "Bình thường, không có gì đặc biệt lắm."
7         ]
8
9     model.eval()
10    print("==> Test kết quả dự đoán (model chưa được fine-tuned):\n")
11    for i, text in enumerate(samples, 1):
12        inputs = tokenizer(text, return_tensors="pt", truncation=True, max_length=256, padding=True).to(model.device)
13        with torch.no_grad():
14            outputs = model(**inputs)
15            pred_class = torch.argmax(outputs.logits, dim=-1).item()
16            pred_label = model.config.id2label[pred_class]
17            print(f"{i}. \"{text}\"\n→ Dự đoán: {pred_label} (label id: {pred_class})\n")
18
```

```
1 test_output_before_training(model, tokenizer)
```

```
==> Test kết quả dự đoán (model chưa được fine-tuned):
```

1. "Giảng viên giảng bài dễ hiểu và nhiệt tình."
→ Dự đoán: tiêu_cực (label id: 0)
2. "Em cảm thấy rất áp lực với lịch học dày đặc."
→ Dự đoán: tích_cực (label id: 2)
3. "Bình thường, không có gì đặc biệt lắm."
→ Dự đoán: tiêu_cực (label id: 0)

Weak performance -> Need fine-tuning.

Supervised Fine-tuning

❖ Step 5: Training

```
1 data_collator = DataCollatorWithPadding(  
2     tokenizer=tokenizer,  
3     pad_to_multiple_of=8,  
4     return_tensors="pt"  
5 )
```

```
1 def compute_metrics(p):  
2     predictions, labels = p  
3     preds = np.argmax(predictions, axis=1)  
4     return {"accuracy": accuracy_score(labels, preds)}
```

```
1 class LossLoggerCallback(TrainerCallback):  
2     def on_log(self, args, state, control, logs=None, **kwargs):  
3         if logs and "loss" in logs:  
4             print(f"[Step {state.global_step:>5}] Loss: {logs['loss']:.4f}")
```

```
1 training_args = TrainingArguments(  
2     output_dir=".//results",  
3     eval_strategy="epoch",  
4     per_device_train_batch_size=2,  
5     per_device_eval_batch_size=2,  
6     gradient_accumulation_steps=4,  
7     num_train_epochs=1,  
8     learning_rate=2e-5,  
9     weight_decay=0.01,  
10    fp16=True,  
11    gradient_checkpointing=True,  
12    remove_unused_columns=True,  
13    logging_dir=".//logs",  
14    logging_steps=50,  
15    save_strategy="epoch",  
16    load_best_model_at_end=True,  
17    report_to="none",  
18    optim="paged_adamw_8bit",  
19    label_names=["labels"]  
20 )  
21
```

Supervised Fine-tuning

❖ Step 5: Training

```
22 trainer = Trainer(  
23     model=model,  
24     args=training_args,  
25     train_dataset=tokenized_dataset["train"],  
26     eval_dataset=tokenized_dataset["validation"],  
27     compute_metrics=compute_metrics,  
28     data_collator=data_collator,  
29     callbacks=[LossLoggerCallback()])  
30 )
```

```
1 trainer.train()
```

[1428/1428 43:24, Epoch 0/1]

Epoch	Training Loss	Validation Loss	Accuracy
0	0.316800	0.361126	0.887555
[Step 50]	Loss: 0.8700		
[Step 100]	Loss: 0.8139		
[Step 150]	Loss: 0.7427		
[Step 200]	Loss: 0.6620		
[Step 250]	Loss: 0.5861		
[Step 300]	Loss: 0.5119		
[Step 350]	Loss: 0.4974		
[Step 400]	Loss: 0.3733		
[Step 450]	Loss: 0.3909		
[Step 500]	Loss: 0.3339		
[Step 550]	Loss: 0.4324		
[Step 600]	Loss: 0.3769		
[Step 650]	Loss: 0.3154		
[Step 700]	Loss: 0.4665		
[Step 750]	Loss: 0.3647		
[Step 800]	Loss: 0.3857		
[Step 850]	Loss: 0.3443		
[Step 900]	Loss: 0.3585		
[Step 950]	Loss: 0.3635		
[Step 1000]	Loss: 0.3565		
[Step 1050]	Loss: 0.4169		
[Step 1100]	Loss: 0.3493		
[Step 1150]	Loss: 0.3666		
[Step 1200]	Loss: 0.4196		
[Step 1250]	Loss: 0.4707		
[Step 1300]	Loss: 0.2827		
[Step 1350]	Loss: 0.3484		
[Step 1400]	Loss: 0.3168		



Supervised Fine-tuning

❖ Step 6: Evaluate on test set

```
1 test_dataset = tokenized_dataset["test"]
2 print("→ Test set:", len(test_dataset), "samples")
```

→ Test set: 3166 samples

```
1 test_results = trainer.evaluate(eval_dataset=test_dataset)
2 print("\nEvaluate on test set:")
3 print({k: round(v, 4) for k, v in test_results.items()})
```

[1583/1583 04:01]

Evaluate on test set:

```
{'eval_loss': 0.4137, 'eval_accuracy': 0.8784, 'eval_runtime': 241.5352, 'eval_samples_per_second': 13.108, 'eval_steps_per_second': 6.554, 'epoch': 0.9998}
```

Supervised Fine-tuning

❖ Step 6: Evaluate on test set

```
1 test_samples = [
2     "Giảng viên giảng bài rất dễ hiểu, em cảm thấy hứng thú với môn học.",
3     "Lịch học quá dày, em cảm thấy rất mệt mỏi và không tiếp thu được bài.",
4     "Em thấy môn học này khá ổn, không có gì đặc biệt.",
5     "Thầy cô tận tình, tuy nhiên hệ thống lớp học trực tuyến hay bị lỗi.",
6     "Môn này khó quá, em học mãi không hiểu gì hết.",
7     "Quá tuyệt vời, không thể nào tốt hơn"
8 ]
9
10 print("\nKết quả phân loại cảm xúc:")
11 model.eval()
12 for i, text in enumerate(test_samples, 1):
13     inputs = tokenizer(text, return_tensors="pt", truncation=True, max_length=256).to(model.device)
14     with torch.no_grad():
15         outputs = model(**inputs)
16     predicted_class = torch.argmax(outputs.logits).item()
17     predicted_label = model.config.id2label[predicted_class]
18     print(f"{i}. {text}\n→ {predicted_label}\n")
```

Kết quả phân loại cảm xúc:

1. Giảng viên giảng bài rất dễ hiểu, em cảm thấy hứng thú với môn học.
→ tích_cực
2. Lịch học quá dày, em cảm thấy rất mệt mỏi và không tiếp thu được bài.
→ tiêu_cực
3. Em thấy môn học này khá ổn, không có gì đặc biệt.
→ tiêu_cực
4. Thầy cô tận tình, tuy nhiên hệ thống lớp học trực tuyến hay bị lỗi.
→ tiêu_cực
5. Môn này khó quá, em học mãi không hiểu gì hết.
→ tiêu_cực
6. Quá tuyệt vời, không thể nào tốt hơn
→ tích_cực

Supervised Fine-tuning

❖ Step 7: Save finetuned model

```
1 model_save_path = "./fine_tuned_model"  
2 model.save_pretrained(model_save_path, safe_serialization=True)  
3 tokenizer.save_pretrained(model_save_path)  
  
('./fine_tuned_model/tokenizer_config.json',  
 './fine_tuned_model/special_tokens_map.json',  
 './fine_tuned_model/chat_template.jinja',  
 './fine_tuned_model/tokenizer.model',  
 './fine_tuned_model/added_tokens.json',  
 './fine_tuned_model/tokenizer.json')
```

Supervised Fine-tuning

❖ Step 8: Inference

```
1 model_save_path = "./fine_tuned_model"
2
3 bnb_config = BitsAndBytesConfig(
4     load_in_4bit=True,
5     bnb_4bit_use_double_quant=True,
6     bnb_4bit_quant_type="nf4",
7     bnb_4bit_compute_dtype=torch.float16
8 )
9
10 base_model = AutoModelForSequenceClassification.from_pretrained(
11     "TinyLlama/TinyLlama-1.1B-Chat-v1.0",
12     quantization_config=bnb_config,
13     num_labels=3,
14     id2label={0: "tiêu_cực", 1: "trung_lập", 2: "tích_cực"},
15     device_map="auto"
16 )
17
18 model = PeftModel.from_pretrained(base_model, model_save_path)
19 model = model.merge_and_unload()
20
21 tokenizer = AutoTokenizer.from_pretrained(model_save_path)
```

```
1 def predict_sentiment(text, model, tokenizer):
2     inputs = tokenizer(
3         text,
4         truncation=True,
5         max_length=256,
6         padding=True,
7         return_tensors="pt"
8     ).to(model.device)
9
10    model.eval()
11    with torch.no_grad():
12        outputs = model(**inputs)
13
14    probabilities = torch.nn.functional.softmax(outputs.logits, dim=-1)
15    predicted_class = torch.argmax(probabilities).item()
16    predicted_label = model.config.id2label[predicted_class]
17    confidence = probabilities[0][predicted_class].item()
18
19    return {
20        "text": text,
21        "label": predicted_label,
22        "confidence": round(confidence, 4),
23        "probabilities": {
24            "tiêu_cực": round(probabilities[0][0].item(), 4),
25            "trung_lập": round(probabilities[0][1].item(), 4),
26            "tích_cực": round(probabilities[0][2].item(), 4)
27        }
28    }
```

Supervised Fine-tuning

❖ Step 8: Inference

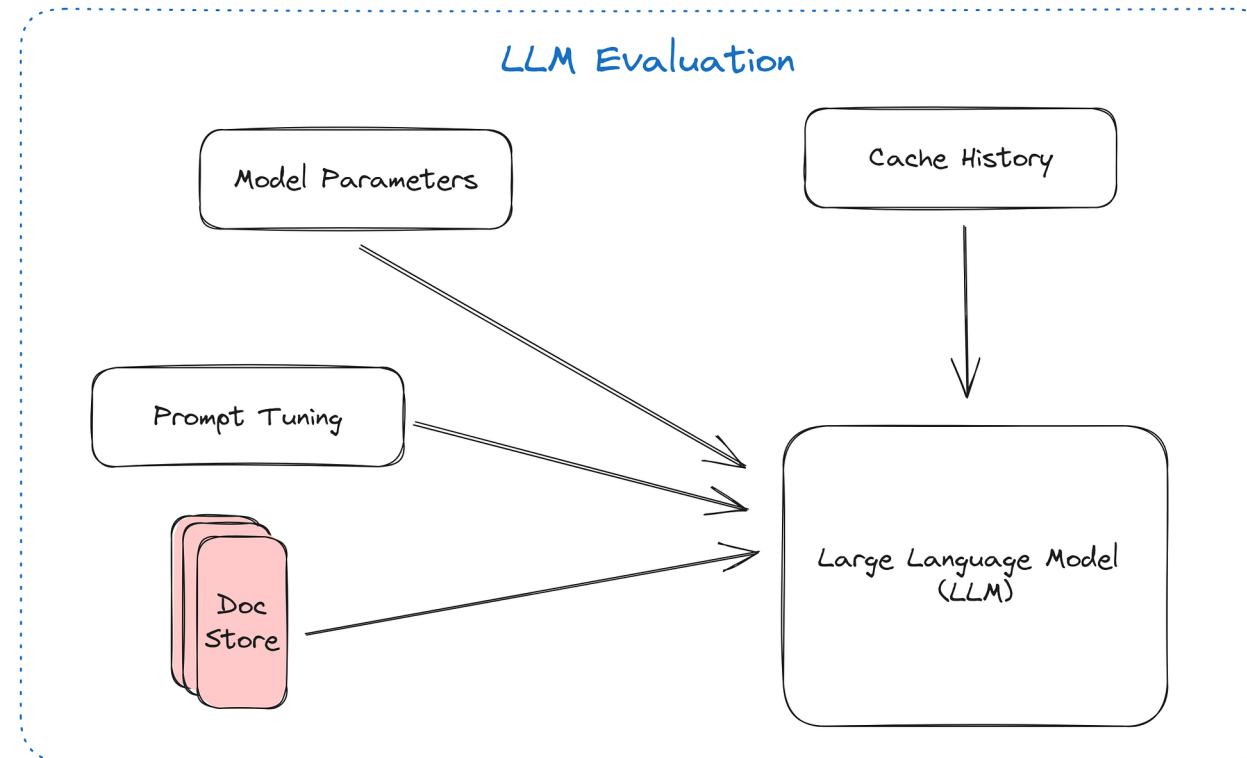
```
1 test_samples = [  
2     "Giáo viên dạy rất nhiệt tình và dễ hiểu",  
3     "Phòng học quá chật hẹp và thiếu trang thiết bị",  
4     "Môn học này khá thú vị nhưng lượng bài tập hơi nhiều",  
5     "Em thấy bình thường",  
6     "Giảng viên nói về môn học này, nghe như đang kể chuyện cổ tích, khó tin nhưng thú vị.",  
7     "Cơ sở vật chất tuyệt vời, chỉ thiếu mỗi việc có thể đưa tôi lên mặt trăng.",  
8     "Môn học này, tôi chắc chắn là mình đã học, nhưng liệu có thể nhớ được không thì chưa biết."  
9 ]
```

- Văn bản: Giáo viên dạy rất nhiệt tình và dễ hiểu
Nhận dự đoán: tích_cực (Độ tin cậy: 91.02%)
Phân phối xác suất: {'tiêu_cực': 0.0522, 'trung_lập': 0.0378, 'tích_cực': 0.9102}
- Văn bản: Phòng học quá chật hẹp và thiếu trang thiết bị
Nhận dự đoán: tiêu_cực (Độ tin cậy: 84.91%)
Phân phối xác suất: {'tiêu_cực': 0.8491, 'trung_lập': 0.0051, 'tích_cực': 0.1458}
- Văn bản: Môn học này khá thú vị nhưng lượng bài tập hơi nhiều
Nhận dự đoán: tích_cực (Độ tin cậy: 83.20%)
Phân phối xác suất: {'tiêu_cực': 0.1343, 'trung_lập': 0.0337, 'tích_cực': 0.832}
- Văn bản: Em thấy bình thường
Nhận dự đoán: tích_cực (Độ tin cậy: 59.67%)
Phân phối xác suất: {'tiêu_cực': 0.2144, 'trung_lập': 0.1892, 'tích_cực': 0.5967}
- Văn bản: Giảng viên nói về môn học này, nghe như đang kể chuyện cổ tích, khó tin nhưng thú vị.
Nhận dự đoán: tiêu_cực (Độ tin cậy: 65.82%)
Phân phối xác suất: {'tiêu_cực': 0.6582, 'trung_lập': 0.0481, 'tích_cực': 0.2935}
- Văn bản: Cơ sở vật chất tuyệt vời, chỉ thiếu mỗi việc có thể đưa tôi lên mặt trăng.
Nhận dự đoán: tiêu_cực (Độ tin cậy: 68.99%)
Phân phối xác suất: {'tiêu_cực': 0.6899, 'trung_lập': 0.1024, 'tích_cực': 0.2074}
- Văn bản: Môn học này, tôi chắc chắn là mình đã học, nhưng liệu có thể nhớ được không thì chưa biết.
Nhận dự đoán: tiêu_cực (Độ tin cậy: 74.61%)
Phân phối xác suất: {'tiêu_cực': 0.7461, 'trung_lập': 0.0482, 'tích_cực': 0.2057}

Model performs better than before it was fine-tuned.

Supervised Fine-tuning

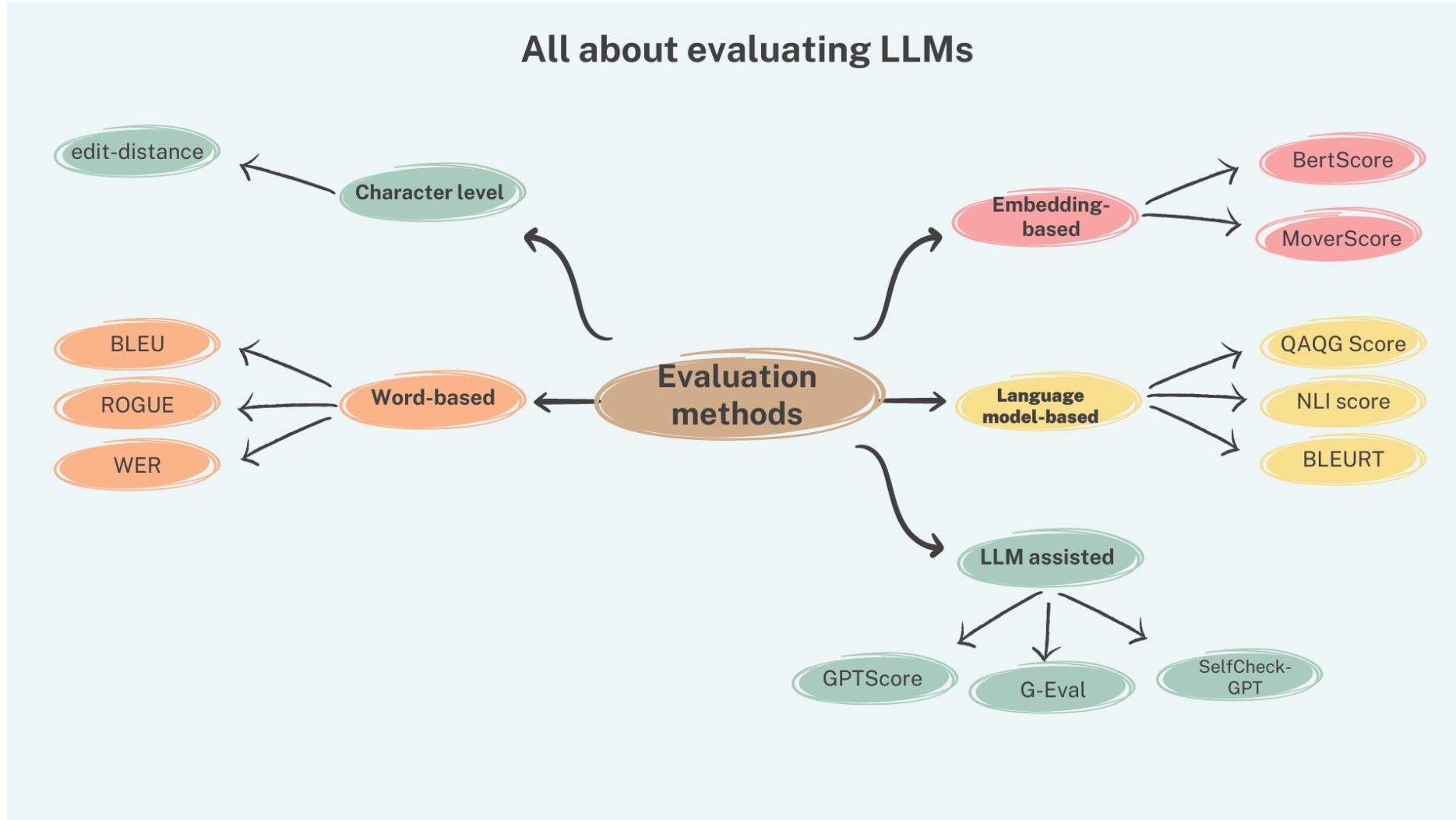
❖ How to evaluate LLMs?



How to efficiently evaluate the performance of LLMs?

Supervised Fine-tuning

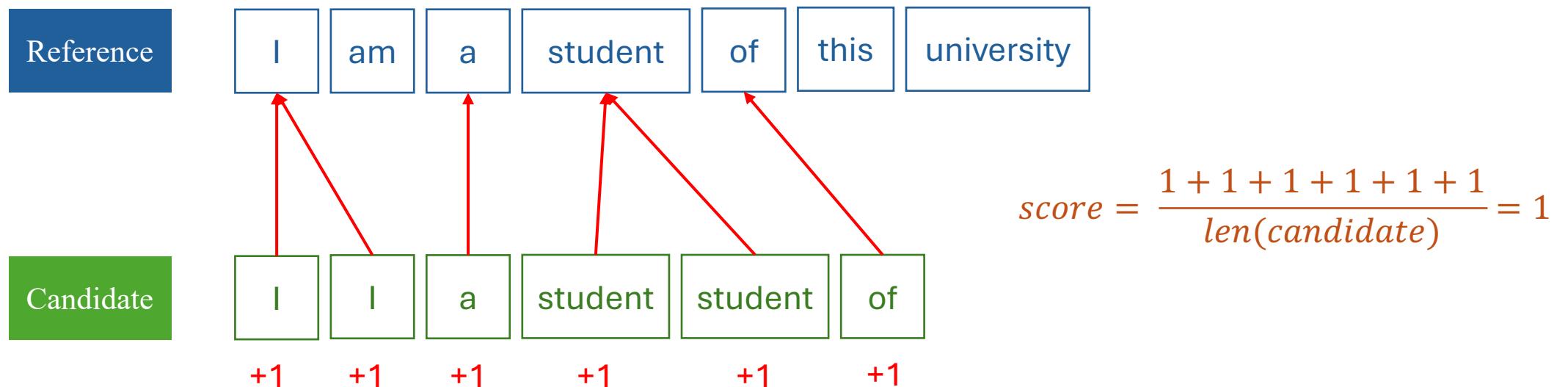
❖ How to evaluate LLMs?



Supervised Fine-tuning

❖ BLEU score

BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated.

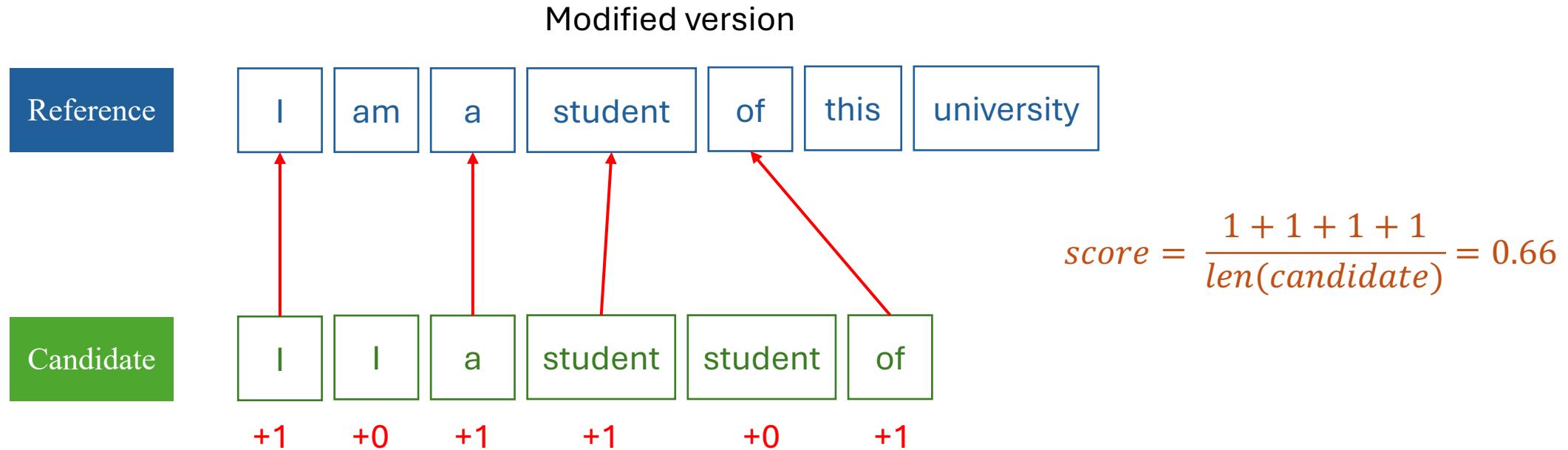


BLEU score is:

How **many words** from the **candidate** appear in the **reference**?

Supervised Fine-tuning

❖ BLEU score



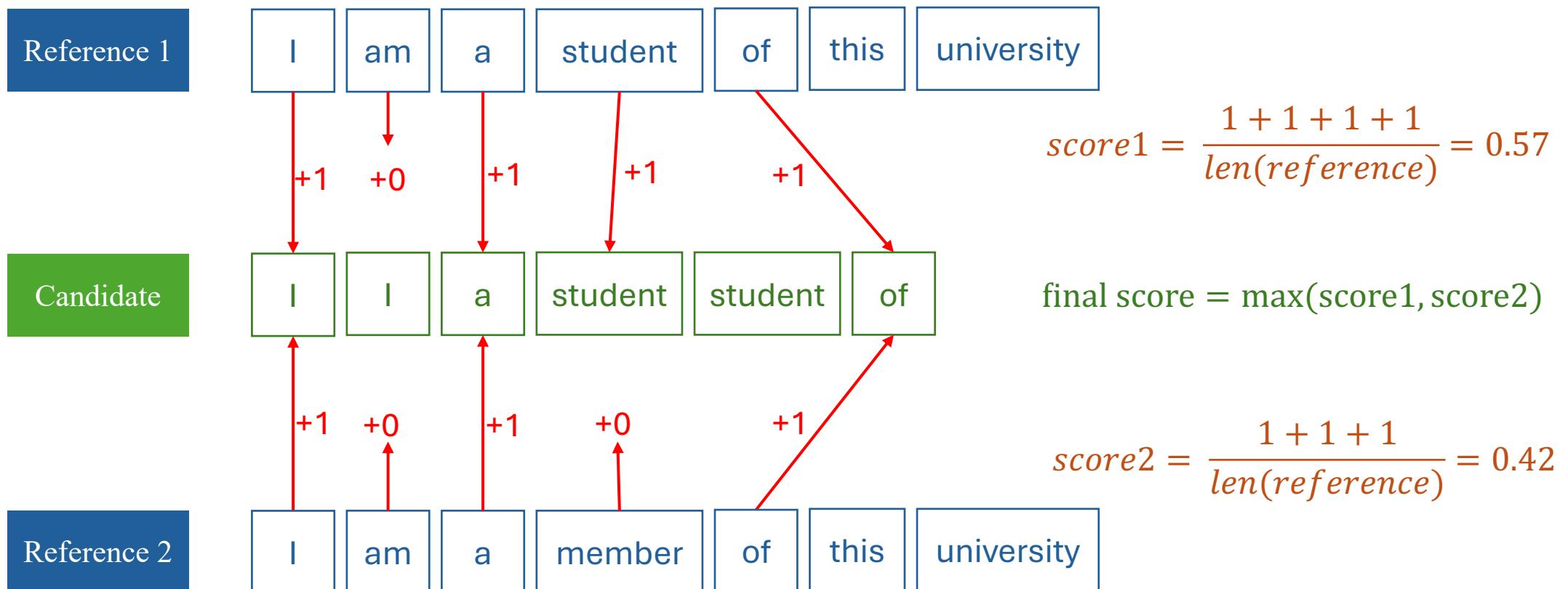
Cons:

- don't consider semantic meaning.
- don't consider sentence structure.

Supervised Fine-tuning

❖ ROUGE score

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics for evaluating text generation models (summarization or machine translation).

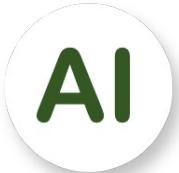


Supervised Fine-tuning

❖ Other metrics?

Rank	Type	Model	Average	IFEval	BBH	MATH	GPQA	MUSR	MMLU-PRO	CO ₂ Cost
1	◆	MaziarPanahi/calme-3.2-instruct-78b	52.08 %	80.63 %	62.61 %	40.33 %	20.36 %	38.53 %	70.03 %	66.01 kg
2	...	MaziarPanahi/calme-3.1-instruct-78b	51.29 %	81.36 %	62.41 %	39.27 %	19.46 %	36.50 %	68.72 %	64.44 kg
3	...	dfurman/CalmeRys-78B-Orpo-v0.1	51.23 %	81.63 %	61.92 %	40.63 %	20.02 %	36.37 %	66.80 %	25.99 kg
4	...	MaziarPanahi/calme-2.4-rys-78b	50.77 %	80.11 %	62.16 %	40.71 %	20.36 %	34.57 %	66.69 %	25.95 kg
5	◆	huihui-ai/Qwen2.5-72B-Instruct-abliterated	48.11 %	85.93 %	60.49 %	60.12 %	19.35 %	12.34 %	50.41 %	76.77 kg
6	...	Qwen/Qwen2.5-72B-Instruct	47.98 %	86.38 %	61.87 %	59.82 %	16.67 %	11.74 %	51.40 %	47.65 kg
7	...	MaziarPanahi/calme-2.1-qwen2.5-72b	47.86 %	86.62 %	61.66 %	59.14 %	15.10 %	13.30 %	51.32 %	29.50 kg
8	◆	newsbang/Homer-v1.0-Qwen2.5-72B	47.46 %	76.28 %	62.27 %	49.02 %	22.15 %	17.90 %	57.17 %	29.55 kg
9	...	ehristoforu/qwen2.5-test-32b-it	47.37 %	78.89 %	58.28 %	59.74 %	15.21 %	19.13 %	52.95 %	29.54 kg
10	◆	Saxo/Linkbricks-Horizon-AI-Avengers-V1-32B	47.34 %	79.72 %	57.63 %	60.27 %	14.99 %	18.16 %	53.25 %	7.95 kg
11	...	MaziarPanahi/calme-2.2-qwen2.5-72b	47.22 %	84.77 %	61.80 %	58.91 %	14.54 %	12.02 %	51.31 %	28.52 kg
12	...	fluently-lm/FluentlyLM-Prinum	47.22 %	80.90 %	59.48 %	54.00 %	18.23 %	17.26 %	53.42 %	21.25 kg
13	?	JungZoona/T3Q-Qwen2.5-14B-Instruct-1M-e3	47.09 %	73.24 %	65.47 %	28.63 %	22.26 %	38.69 %	54.27 %	1.40 kg
14	◆	JungZoona/T3Q-qwen2.5-14b-v1.0-e3	47.09 %	73.24 %	65.47 %	28.63 %	22.26 %	38.69 %	54.27 %	1.56 kg

Evaluate LLMs on datasets built specifically for some capabilities.



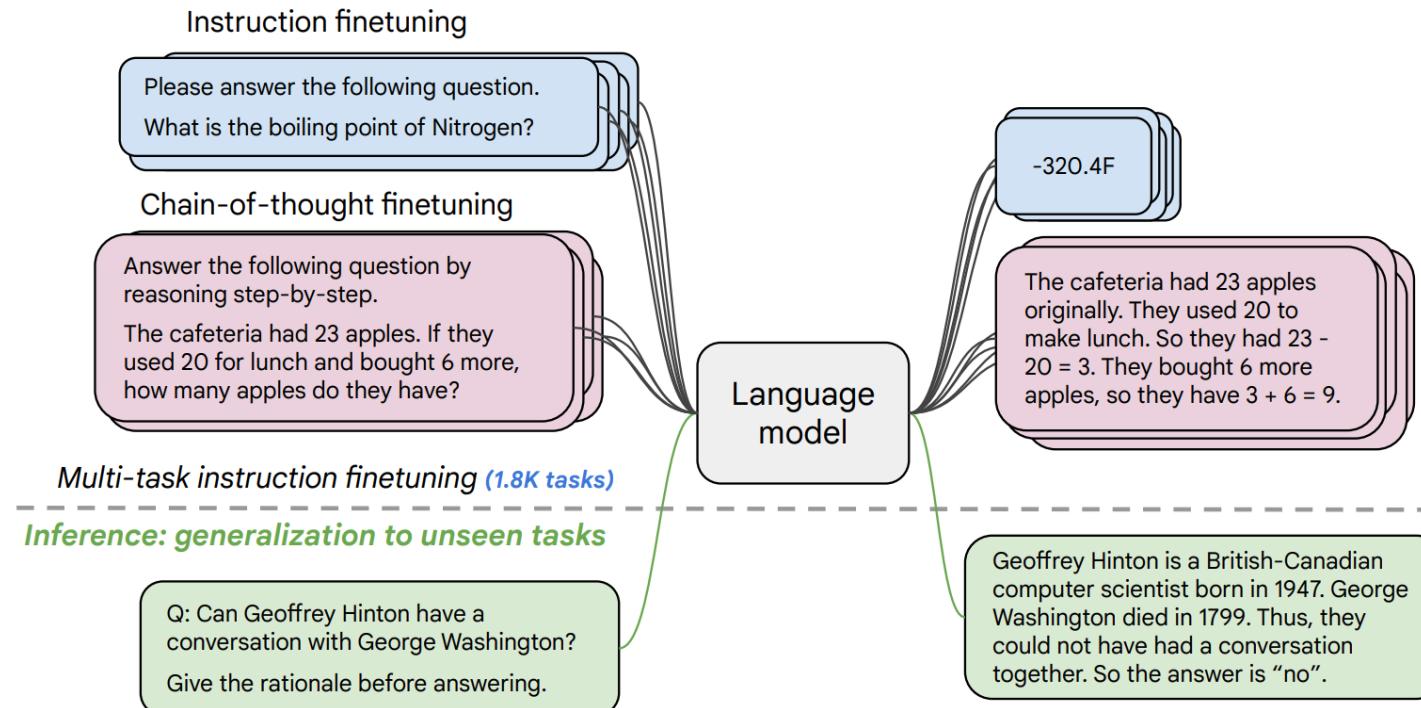
AI

AI VIET NAM
@aivietnam.edu.vn

Instruction Tuning

Instruction Tuning

❖ Introduction

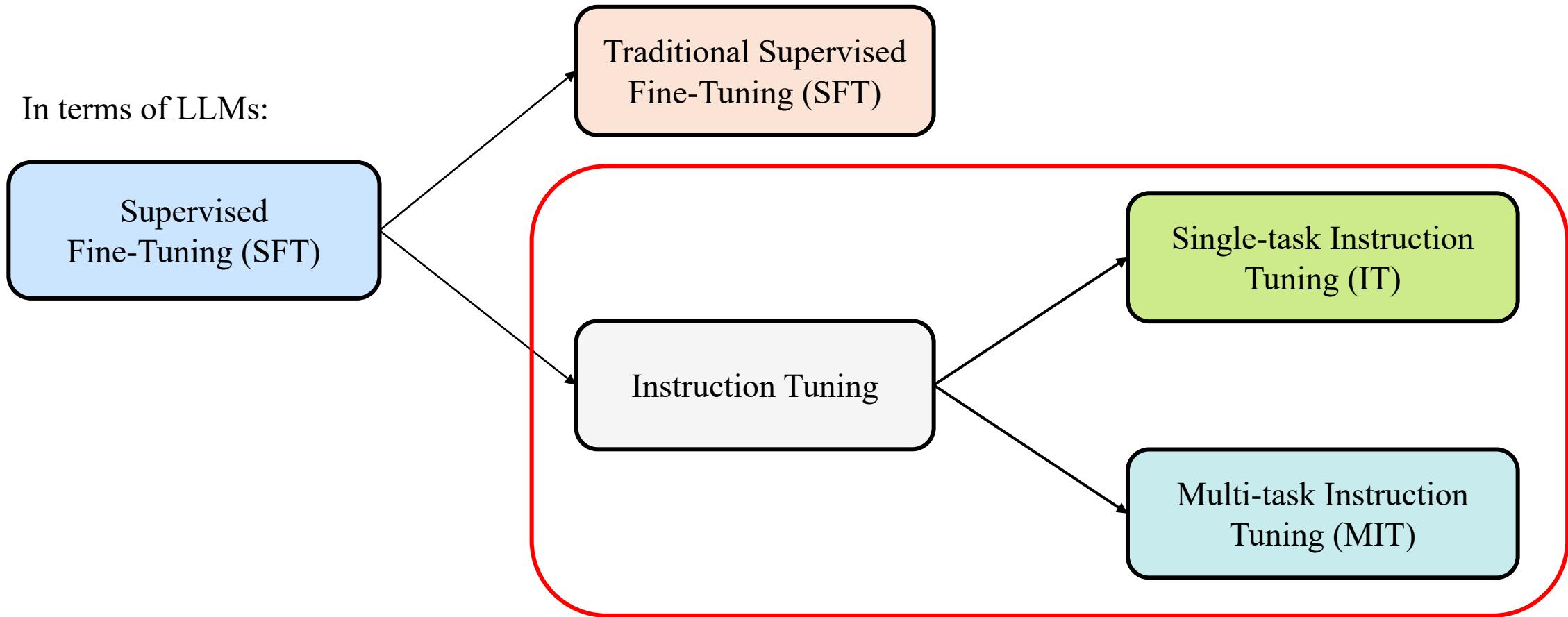


Instruction Tuning in LLMs is a training method aimed at enhancing the model's ability to understand and execute natural language instructions.

Instruction Tuning

❖ Introduction

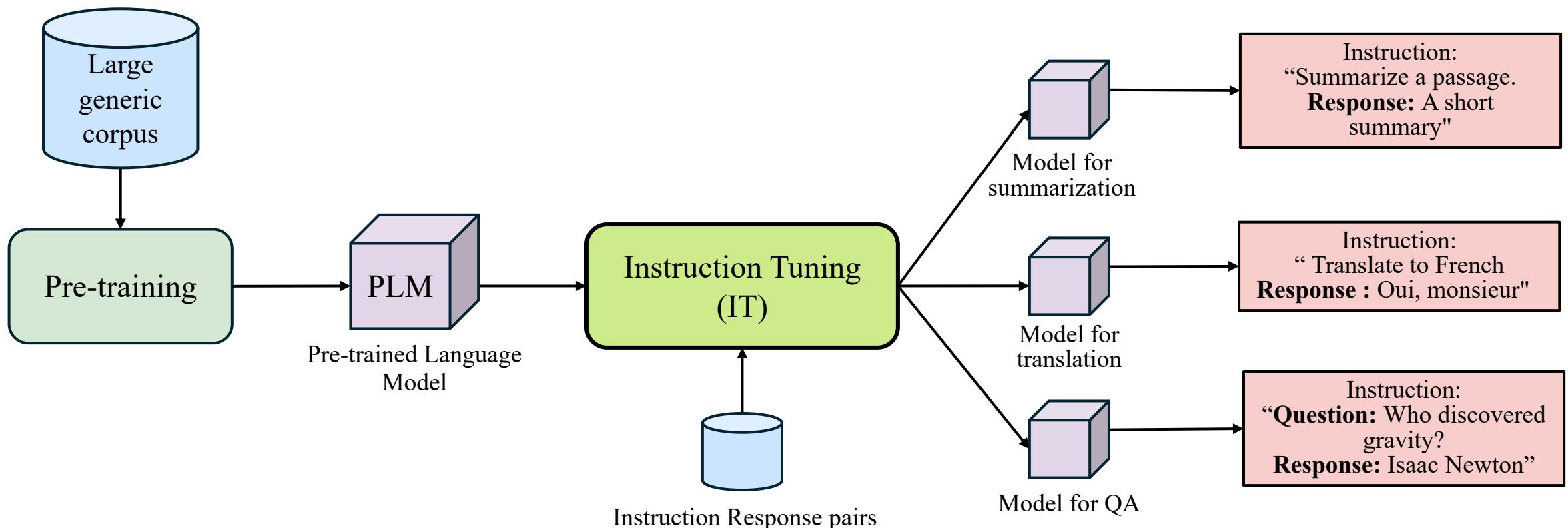
In terms of LLMs:



Instruction Tuning

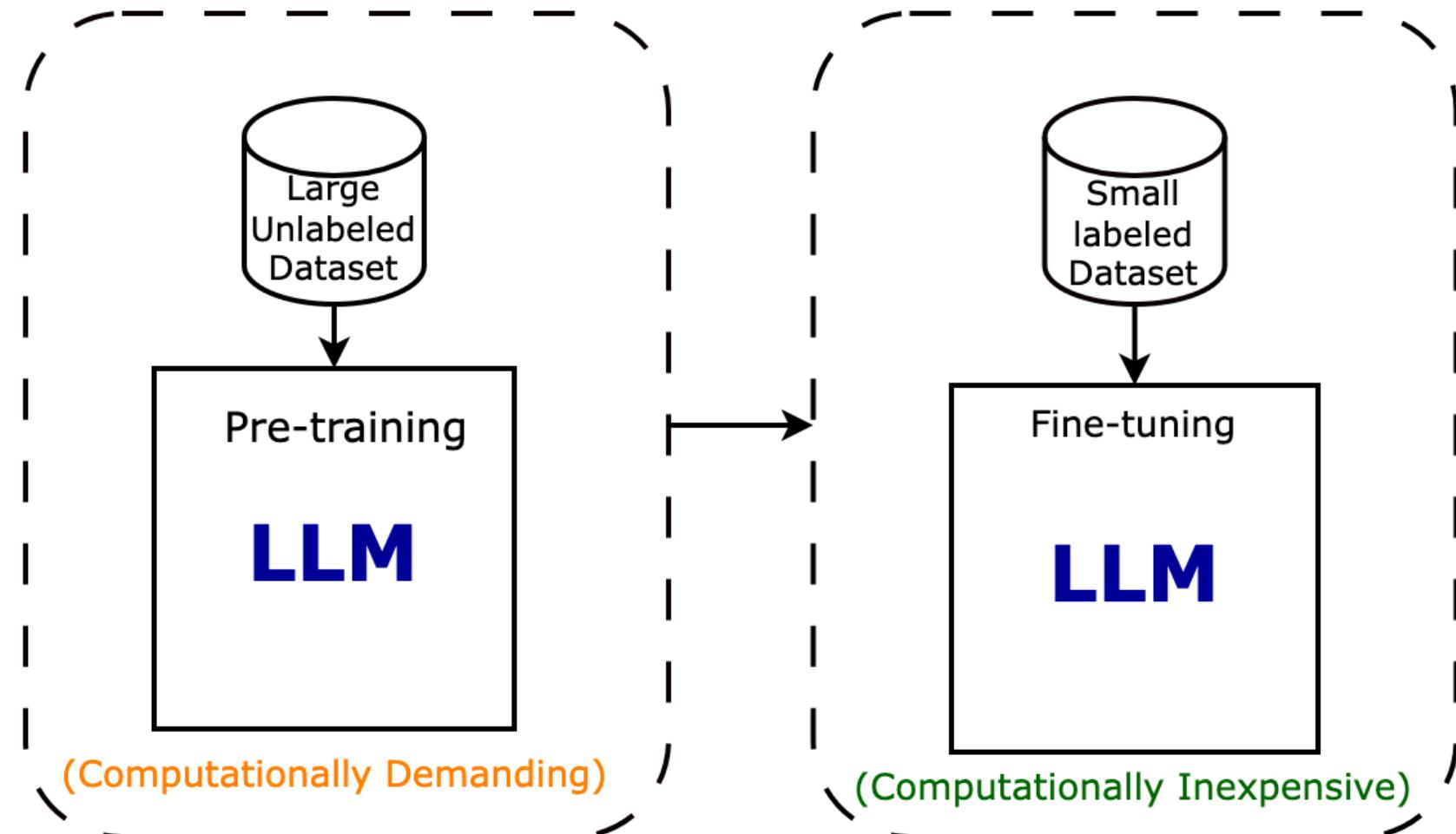
❖ Single-task Instruction Tuning (IT)

Single-task Instruction Tuning or Instruction Tuning (IT) for short teaches a model to follow natural language instructions by training it on diverse prompts and corresponding responses. This helps the model better understand and execute user intent.



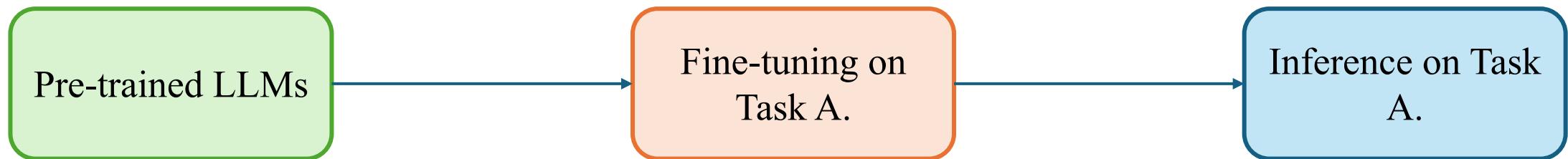
Instruction Tuning

❖ Applications



Instruction Tuning

❖ Applications

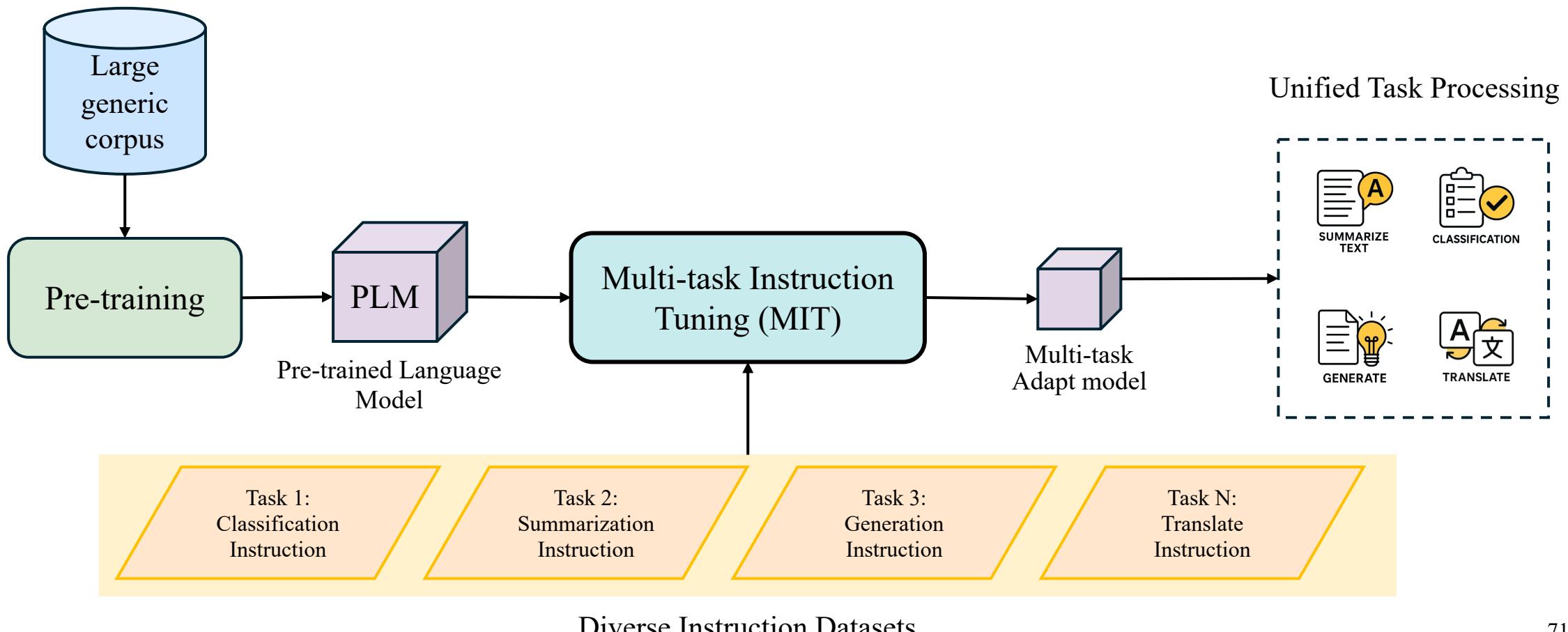


Fine-tuning an LLM to perform on a single task.

Instruction Tuning

❖ Multi-task Instruction Tuning (MIT)

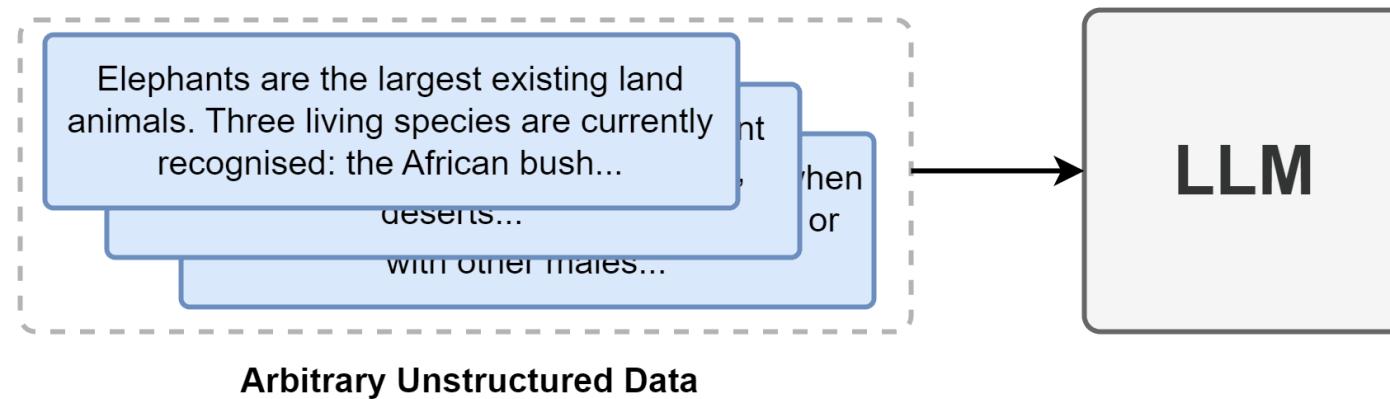
Multi-task Instruction Tuning (MIT) extends instruction tuning by exposing the model to a wide range of instructions from various domains and tasks. It enhances the model's ability to generalize and perform well across diverse scenarios.



Instruction Tuning

❖ Training LLMs

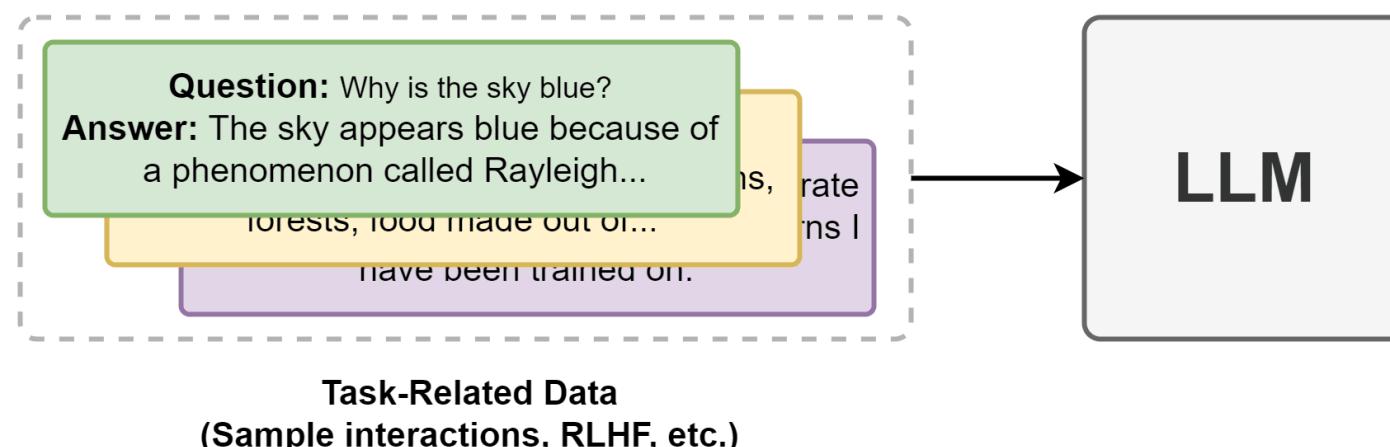
Stage 1: Pretraining (Expensive, Unaligned)



LLM

Arbitrary Unstructured Data

Stage 2: Instruction Tuning (Cheap, Where Alignment Happens)



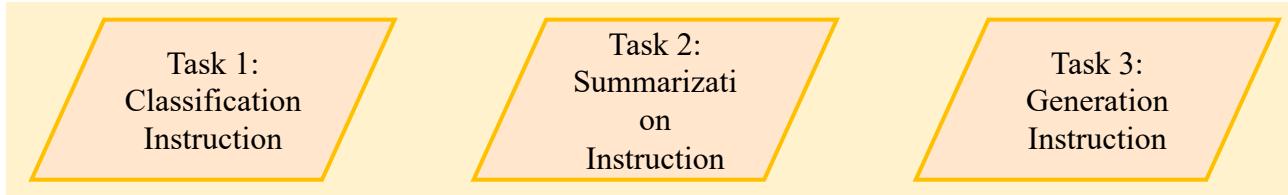
LLM

Task-Related Data

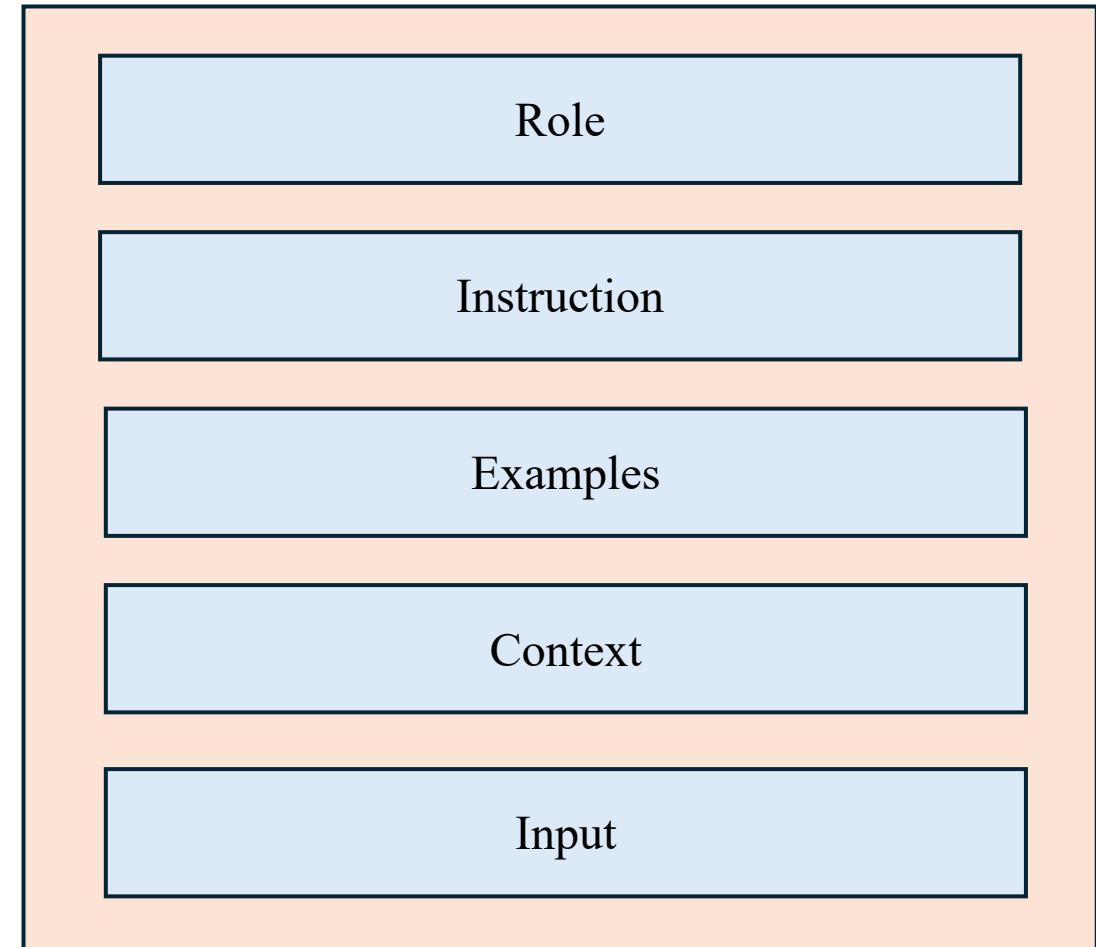
(Sample interactions, RLHF, etc.)

Instruction Tuning

❖ Input format to LLMs



Given a dataset of a single task, we need to organize it to a instruction-like prompt format for each sample.



Instruction Tuning

❖ Input format to LLMs

Role

Instruction

Examples

Context

Input

You are a helpful assistant who explains technical concepts in simple language.

Task: Summarize the following paragraph in one sentence.

Example:

Input: "Artificial intelligence enables machines to mimic human intelligence and behavior."

Output: "AI lets machines act like humans."

Context: The paragraph is from a high school science textbook aimed at beginners.

Input: "Machine learning is a subset of AI that allows systems to learn and improve from experience without being explicitly programmed."

Instruction Tuning

❖ Use case: SAT reading question solver



Please help me answer the SAT Reading question below and explain your choice.

Passage: "Elizabeth... her spirits were in a high flutter."

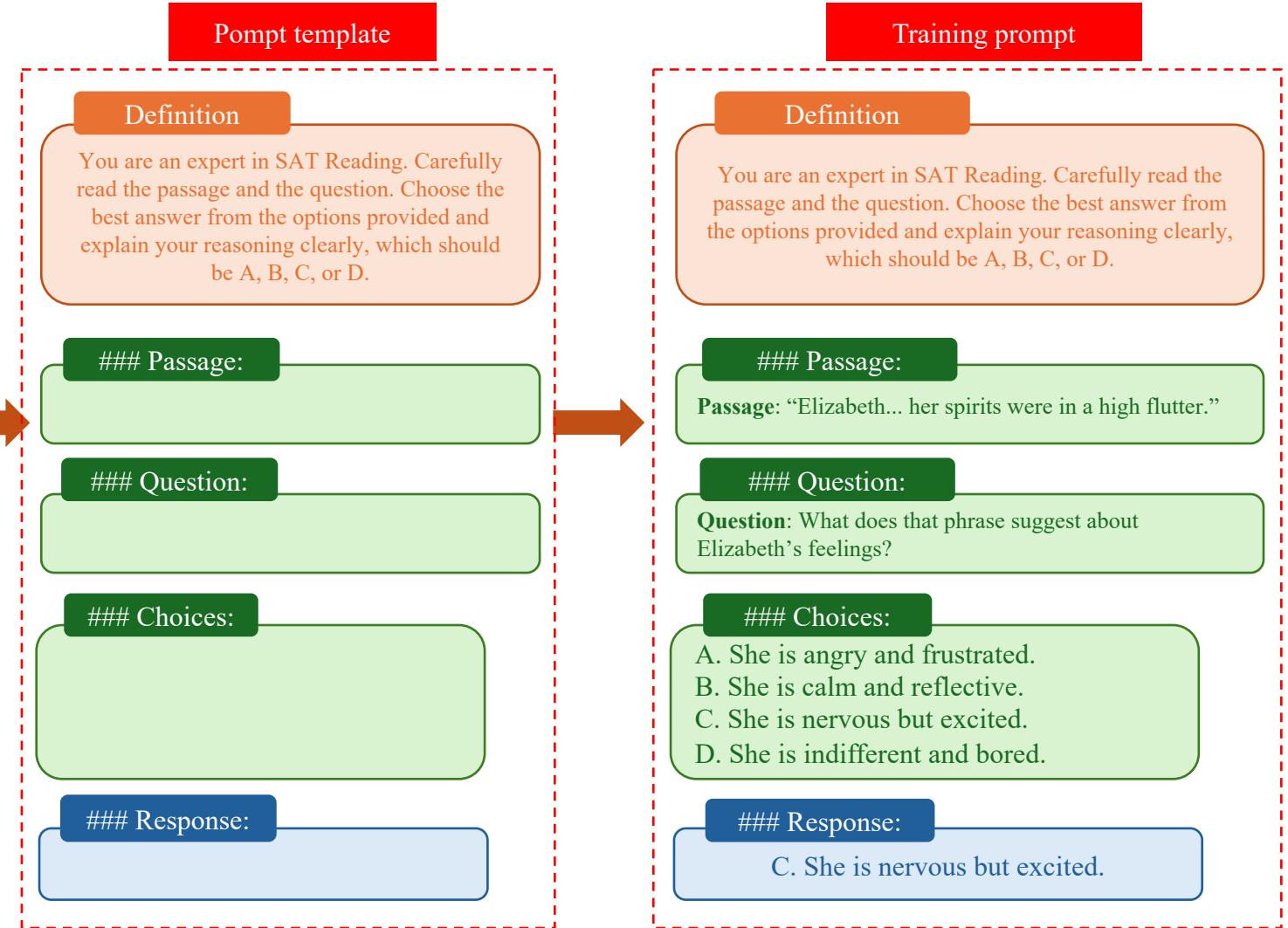
Question: What does that phrase suggest about Elizabeth's feelings?

- A. She is angry and frustrated. C. She is nervous but excited.
- B. She is calm and reflective. D. She is indifferent and bored.

Modify Prompt

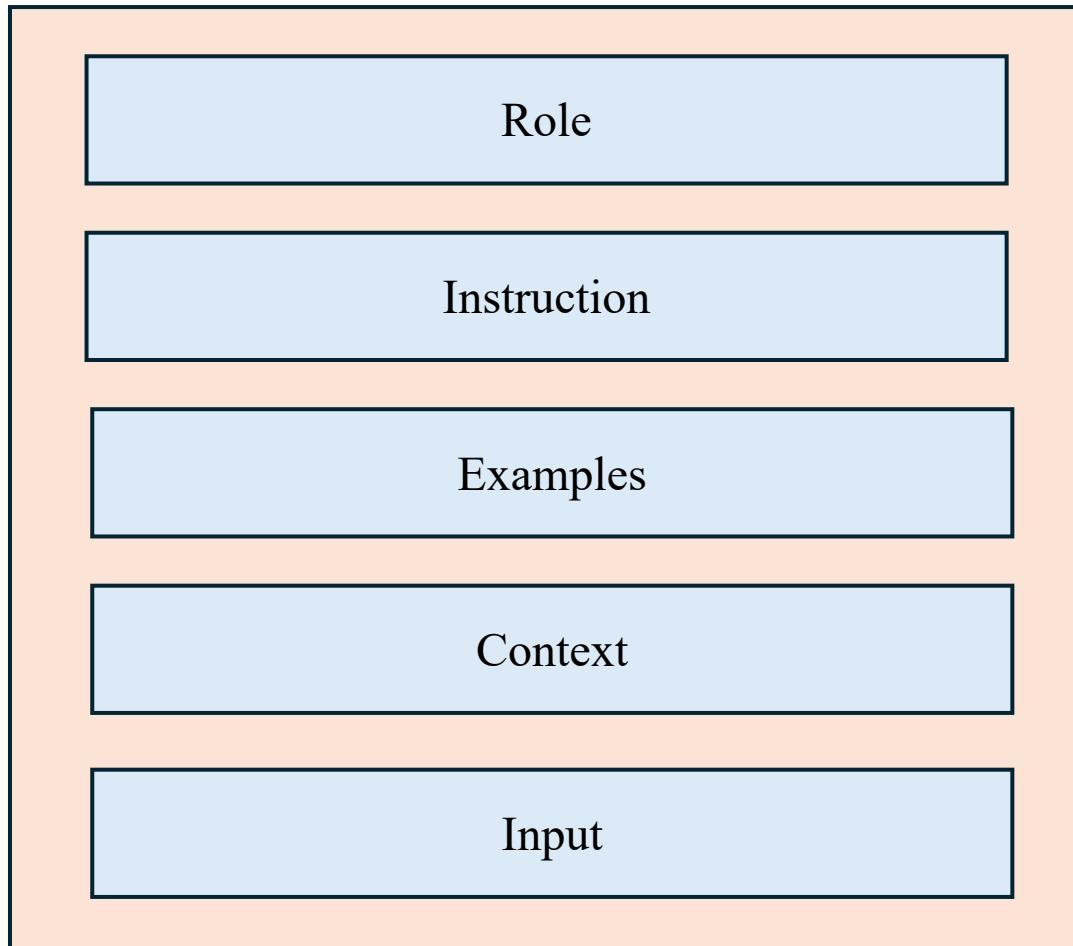


C. She is nervous but excited.



Instruction Tuning

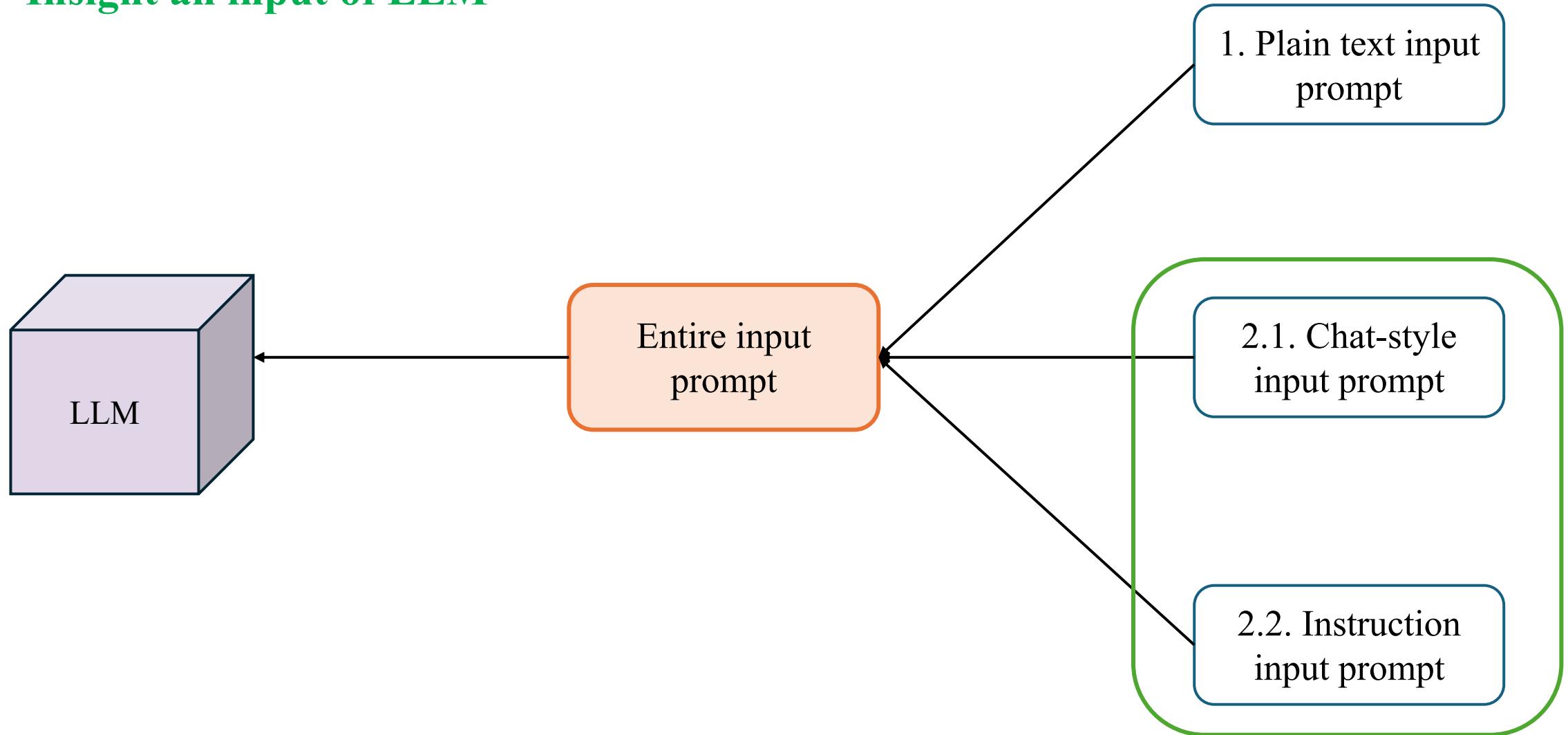
❖ How to put this data into an LLM?



Given an instruction prompt, how can we put this data into the input of an LLM?

Instruction Tuning

❖ Insight an input of LLM





Instruction Tuning

❖ Insight an input of LLM

{"role": "system", "content": "You are a helpful assistant that summarizes content clearly."},

{"role": "user", "content": "Please summarize the following:\n\nMachine learning is a field of AI that allows computers to learn from data without being explicitly programmed."},

{"role": "assistant", "content": "Sure! Machine learning helps computers learn from data automatically, without explicit programming."}

<s>[INST] <<SYS>>

You are a helpful assistant that summarizes content clearly.

<</SYS>>

Please summarize the following:

Machine learning is a field of AI that allows computers to learn from data without being explicitly programmed.

[/INST] Sure! Machine learning helps computers learn from data automatically, without explicit programming. </s>

Instruction:

Summarize the following paragraph.

Input:

Machine learning is a field of AI that allows computers to learn from data without being explicitly programmed.

Response:

Sure! Machine learning helps computers learn from data automatically, without explicit programming.

Instruction Tuning

❖ Chat-style model: Conversation

```
[  
  {  
    "role": "system",  
    "content": "You are a helpful assistant that summarizes content clearly."  
  },  
  {  
    "role": "user",  
    "content": "Please summarize the following:\n\nMachine learning is a field of AI that allows computers to learn from data without being explicitly programmed."  
  },  
  {  
    "role": "assistant",  
    "content": "Machine learning helps computers learn from data automatically, without needing explicit instructions."  
  }]
```

Single-turn

```
[  
  {  
    "role": "system",  
    "content": "You are a helpful assistant that summarizes content clearly."  
  },  
  {  
    "role": "user",  
    "content": "Please summarize the following:\n\nMachine learning is a field of AI that allows computers to learn from data without being explicitly programmed."  
  },  
  {  
    "role": "assistant",  
    "content": "Machine learning helps computers learn from data automatically, without needing explicit instructions."  
  },  
  {  
    "role": "user",  
    "content": "Can you also summarize this?\n\nDeep learning is a subset of machine learning that uses neural networks with many layers."  
  },  
  {  
    "role": "assistant",  
    "content": "Deep learning is a type of machine learning that uses multi-layered neural networks to learn complex patterns from data."  
  }]
```

Multi-turn



Instruction Tuning

❖ Llama 3.2 Prompt Template

Supported Roles: There are 4 different roles that are supported by Llama text models: system, assistant, user, ipython.

[system, assistant, user, ipython]

Supported Role	Description
system	Sets the context in which to interact with the AI model. It typically includes rules, guidelines, or necessary information that help the model respond effectively.
user	Represents the human interacting with the model. It includes the inputs, commands, and questions to the model.
ipython	A new role introduced in Llama 3.1. Semantically, this role means "tool". This role is used to mark messages with the output of a tool call when sent back to the model from the executor.
assistant	Represents the response generated by the AI model based on the context provided in the system , ipython and user prompts.



Instruction Tuning

❖ Llama 3.2 Prompt Template

Special Tokens	Description
< begin_of_text >	Specifies the start of the prompt.
< end_of_text >	Model will cease to generate more tokens. This token is generated only by the base models.
< finetune_right_pad_id >	This token is used for padding text sequences to the same length in a batch.
< start_header_id >	These tokens enclose the role for a particular message. The possible roles are: [system, user, assistant, and ipython]
< eom_id >	End of message. A message represents a possible stopping point for execution where the model can inform the executor that a tool call needs to be made. This is used for multi-step interactions between the model and any available tools. This token is emitted by the model when the Environment: ipython instruction is used in the system prompt, or if the model calls for a built-in tool.
< eot_id >	End of turn. Represents when the model has determined that it has finished interacting with the user message that initiated its response. This is used in two scenarios: <ul style="list-style-type: none">• at the end of a direct interaction between the model and the user• at the end of multiple interactions between the model and any available tools This token signals to the executor that the model has finished generating a response.
< python_tag >	Special tag used in the model's response to signify a tool call.

Instruction Tuning

❖ Llama 3.2 Prompt Template

Instruct Model Prompt: The format for a regular multi-turn conversation between a user and the model of Llama 3.2.

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

Cutting Knowledge Date: December 2023

Today Date: 23 July 2024

You are a helpful assistant.<|eot_id|>

```
<|start_header_id|>user<|end_header_id|>
```

What is the capital of France?<|eot_id|>

```
<|start_header_id|>assistant<|end_header_id|>
```

The capital of France is Paris.

- Each message role clearly marked with header tokens.
- <|eot_id|> separates each interaction turn.
- System content should define environment, cut-off date, tone, and rules.

Instruction Tuning

❖ Llama 3.2 Use Case: Code Interpreter Prompt

Sample Input:

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

```
Environment: ipython<|eot_id|>
<|start_header_id|>user<|end_header_id|>
```

Write code to check if number is prime, use that to see if the number 7 is prime<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>

Response:

```
<|python_tag|>def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(7)) # Output: True<|eom_id|>
```

- If the system prompt includes Environment: ipython, it enables code interpreter mode.
- The model's response contains a code block wrapped in <|python_tag|>.
- It ends with <|eom_id|>, signaling tool execution is expected before continuing.



Instruction Tuning

❖ Input instruction to Chat-style LLMs

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

Cutting Knowledge Date: December 2023

Today Date: 23 July 2024

You are a helpful assistant.<|eot_id|>

```
<|start_header_id|>user<|end_header_id|>  
{input_data}
```

```
<|start_header_id|>assistant<|end_header_id|>
```

You are a helpful assistant who explains technical concepts in simple language.

Task: Summarize the following paragraph in one sentence.

Example:

Input: "Artificial intelligence enables machines to mimic human intelligence and behavior."

Output: "AI lets machines act like humans."

Context: The paragraph is from a high school science textbook aimed at beginners.

Input: "Machine learning is a subset of AI that allows systems to learn and improve from experience without being explicitly programmed."

Instruction Tuning

❖ Input instruction data to Chat LLMs



Please help me answer the SAT Reading question below and explain your choice.
Passage: "This passage is adapted from Elizabeth Cady Stanton's address to the 1869 Woman Suffrage Convention in Washington, DC..."
Question: Stanton claims that which of the following was a relatively recent historical development?
A. The control of society by men. C. The domination of domestic life by men.
B. The spread of war and injustice. D. The acknowledgement of women's true character.



D. The acknowledgement of women's true character.

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

Cutting Knowledge Date: December 2023
Today Date: 26 Jul 2024

You are a helpful AI assistant developed by Meta. Respond safely and accurately.

<|eot_id|><|start_header_id|>user<|end_header_id|>

Read the passage and answer the question.

Passage:

This passage is adapted from Elizabeth Cady Stanton's address to the 1869 Woman Suffrage Convention in Washington, DC. I urge a sixteenth amendment, because

...

and not with her consent would one drop of blood ever be shed, one life sacrificed in vain.

Question:

Stanton claims that which of the following was a relatively recent historical development?

Choices:

- A) The control of society by men
- B) The spread of war and injustice
- C) The domination of domestic life by men
- D) The acknowledgement of women's true character

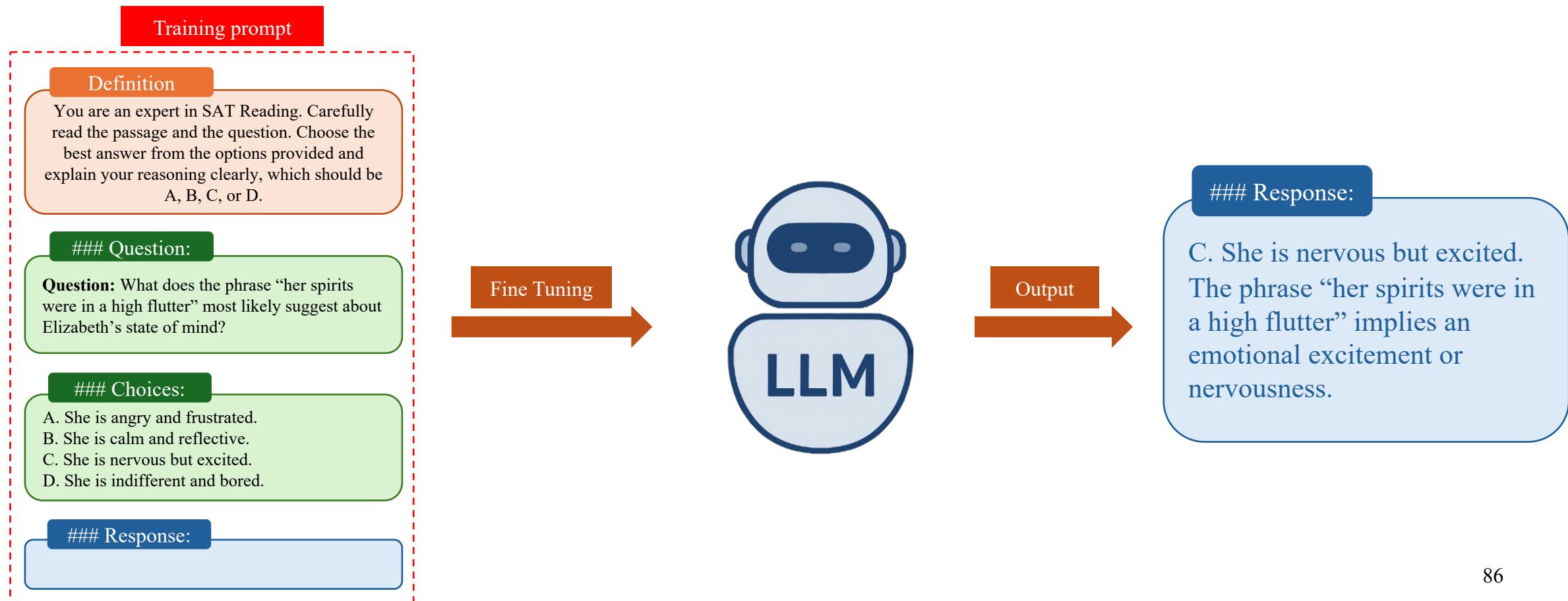
Respond with ONLY the letter and full text of the correct answer.<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>

D) The acknowledgement of women's true character<|eot_id|>

Instruction Tuning

❖ Use case: SAT reading question solver

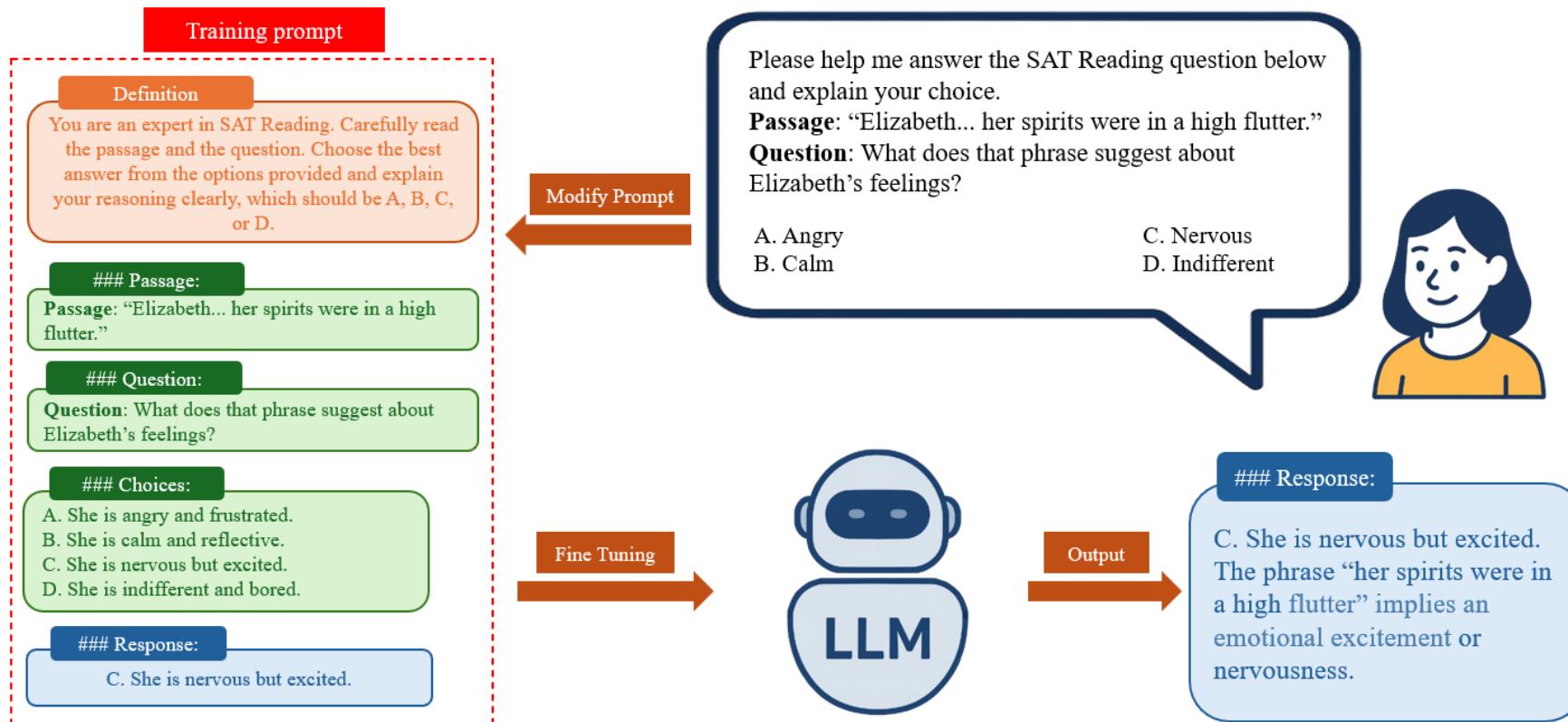
Problem statement: Build a model that can choose the best answer (A, B, C or D) given a multiple choice SAT in reading question.



Instruction Tuning

❖ Problem Description

Description: Build an English reading comprehension chatbot for standardized reading tests (IELTS/SAT) using the Llama 3 LLM.





Instruction Tuning

❖ Problem Description

Description: Build an English reading comprehension chatbot for standardized reading tests (IELTS/SAT) using the Llama 3 LLM.

system

Cutting Knowledge Date: December 2023
Today Date: 26 Jul 2024

You are a helpful AI assistant developed by Meta. Respond safely and accurately.user

Analyze the given passage and question. Choose the best answer from the options below.

Passage:

Anna sat by the window every day, watching the rain fall.
She missed the sunshine, but she never said it out loud.
Instead, she smiled at everyone and kept her feelings to herself.

Question:

What does the passage suggest about Anna?

Choices:

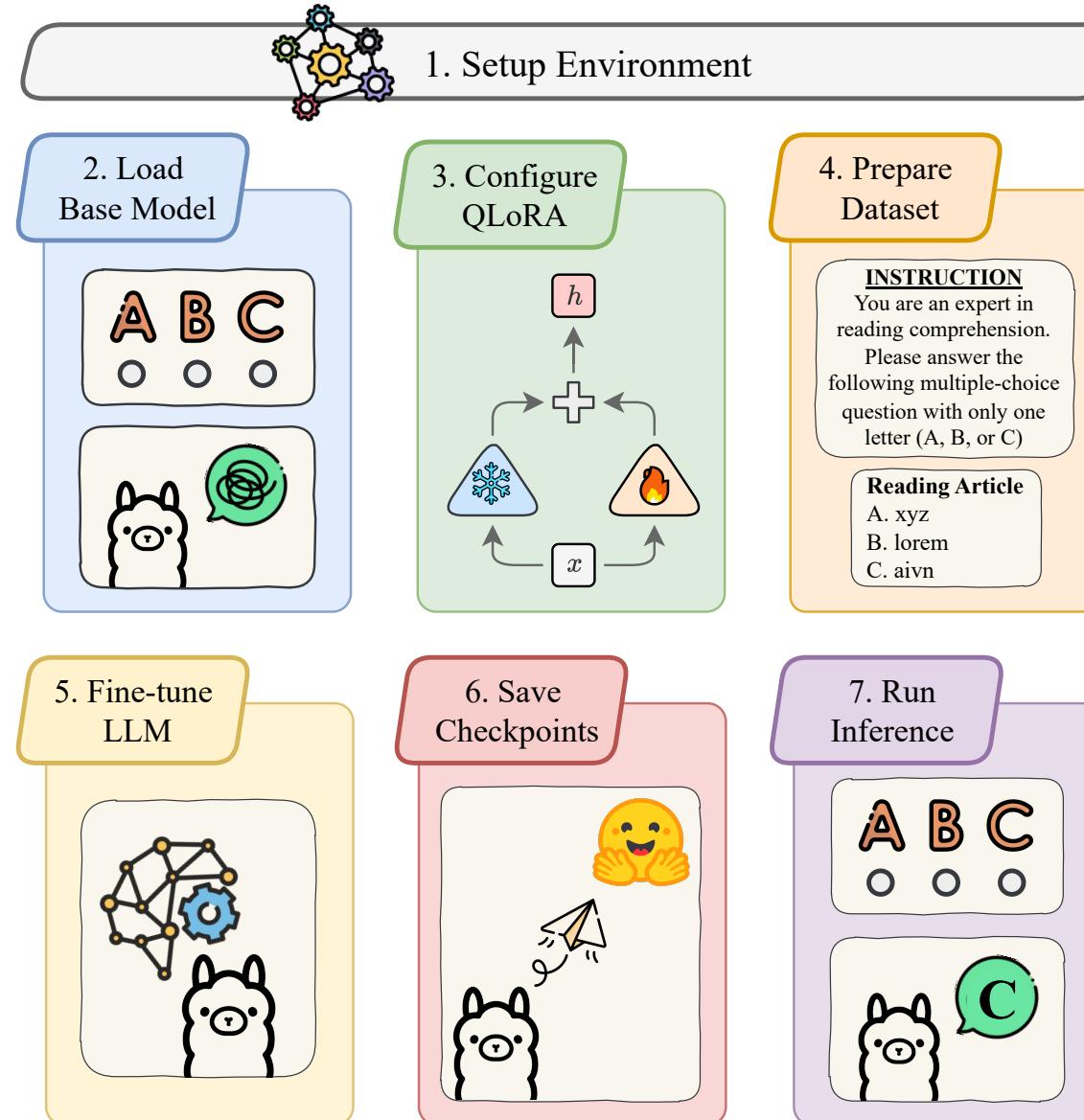
- A) She enjoys rainy days.
- B) She hides her true feelings.
- C) She likes to talk about the weather.
- D) She avoids sitting by the window.

Respond ONLY with the letter and full text of the correct answer choice.assistant

- B) She hides her true feelings.

Instruction Tuning

❖ Problem Pipeline



Instruction Tuning

❖ Step 1: Install and import necessary libraries

```
!pip install -q -U bitsandbytes
!pip install -q -U datasets
!pip install -q -U git+https://github.com/huggingface/transformers.git
!pip install -q -U git+https://github.com/huggingface/peft.git
!pip install -q -U git+https://github.com/huggingface/accelerate.git
!pip install -q -U loralib
!pip install -q -U einops
```

```
1 import os
2 import torch
3 from datasets import load_dataset, Dataset
4 from transformers import (
5     AutoTokenizer, AutoModelForCausalLM,
6     BitsAndBytesConfig, GenerationConfig,
7     TrainingArguments, Trainer, DataCollatorForLanguageModeling
8 )
9 from peft import (
10    LoraConfig, get_peft_model,
11    prepare_model_for_kbit_training
12 )
13 from huggingface_hub import login
14 from sklearn.model_selection import train_test_split
15 from tqdm import tqdm
16 import re
```

Instruction Tuning

❖ Step 2: Load pre-trained model

```
1 from huggingface_hub import login
2
3 # Log in to Hugging Face
4 login(token="hf_                                /GmrZ")
```

```
1 os.environ["CUDA_VISIBLE_DEVICES"] = "0"
2 MODEL_NAME = "meta-llama/Llama-3.2-3B-Instruct"
3
4 bnb_config = BitsAndBytesConfig(
5     load_in_4bit=True,
6     bnb_4bit_use_double_quant=True,
7     bnb_4bit_quant_type="nf4",
8     bnb_4bit_compute_dtype=torch.bfloat16
9 )
10
11 model = AutoModelForCausalLM.from_pretrained(
12     MODEL_NAME,
13     device_map="auto",
14     trust_remote_code=True,
15     quantization_config=bnb_config
16 )
17
18 tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
19 tokenizer.pad_token = tokenizer.eos_token
```

Loading checkpoint shards: 100% |██████████| 2/2 [00:07<00:00, 3.82s/it]



Instruction Tuning

❖ Step 3: Configure LLMs

```
1 model.gradient_checkpointing_enable()
2 model = prepare_model_for_kbit_training(model)

1 config = LoraConfig(
2     r=16,
3     lora_alpha=32,
4     target_modules=[
5         "q_proj",
6         "up_proj",
7         "o_proj",
8         "k_proj",
9         "down_proj",
10        "gate_proj",
11        "v_proj"
12    ],
13     lora_dropout=0.05,
14     bias="none",
15     task_type="CAUSAL_LM"
16 )
17
18 model = get_peft_model(model, config)
```

```
1 def print_trainable_parameters(model):
2     """
3         Prints the number of trainable parameters in the model.
4     """
5     trainable_params = 0
6     all_param = 0
7
8     for _, param in model.named_parameters():
9         all_param += param.numel()
10        if param.requires_grad:
11            trainable_params += param.numel()
12
13    print(
14        f"trainable params: {trainable_params} || all params: {all_param} || trainables%: {100 * trainable_params / all_param}"
15    )
16
17 print_trainable_parameters(model)
```

trainable params: 24313856 || all params: 1827777536 || trainables%: 1.330241537665993

LoRA reduces trainable parameters to just 1.33%, drastically lowering compute and memory costs.

Instruction Tuning

❖ Step 3: Configure LLMs

```
1 from transformers import GenerationConfig
```

```
1 generation_config = GenerationConfig(  
2     max_new_tokens=50,  
3     temperature=0.01,           Low temperature makes output more deterministic (less random).  
4     do_sample=False,  
5     num_return_sequences=1,  
6     pad_token_id=tokenizer.eos_token_id,  
7     eos_token_id=tokenizer.eos_token_id,  
8     repetition_penalty=1.3      Penalizes repeated tokens to reduce redundancy.  
9 )
```

Generation configuration (Will affect the generation results).



Instruction Tuning

❖ Step 4: Test pretrained model performance

```
1 # Llama-3's official system prompt structure
2 LLAMA3_SYSTEM_PROMPT = """You are a helpful AI assistant developed by Meta.
3 | | | | | | | | | | | | Respond safely and accurately."""
```

```
1 # Test pretrained model performance
2 prompt = [
3     {
4         "role": "system",
5         "content": LLAMA3_SYSTEM_PROMPT
6     },
7     {
8         "role": "user",
9         "content": """Analyze the given passage and question.
10 | Choose the best answer from the options below.
11
12 ### Passage:
13 [This passage is adapted from George Eliot, Silas Marner. Originally published
14 in 1861. Silas was a weaver and a notorious miser, but then the gold he had
15 hoarded was stolen. Shortly after, Silas adopted a young child, Eppie, the
16 daughter of an impoverished woman who had died suddenly.
17
18 | | Unlike the gold which needed nothing, and must
19 be worshipped in close-locked solitude—which was
20 hidden away from the daylight, was deaf to the song
21 of birds, and started to no human tones—Eppie was a
```

```
86 Silas puzzled on such occasions by the incompatible
87 demands of love.]
```

88

89 ### Question:

90 Which statement best describes a technique the narrator uses to represent

91 Silas's character before he adopted Eppie?

92

93 ### Choices:

94 A) The narrator emphasizes Silas's former obsession with wealth by depicting his

95 | gold as requiring certain behaviors on his part.

96 B) The narrator underscores Silas's former greed by describing his gold as

97 | seeming to reproduce on its own.

98 C) The narrator hints at Silas's former antisocial attitude by contrasting his

99 | present behavior toward his neighbors with his past behavior toward them.

100 D) The narrator demonstrates Silas's former lack of self-awareness by implying

101 | that he is unable to recall life before Eppie.

102

103 Respond ONLY with the letter and full text of the correct answer choice."""
104 | | }

105]



Instruction Tuning

❖ Step 4: Test pretrained model performance

```
1 device = 'cuda' if torch.cuda.is_available() else 'cpu'
2
3 # Apply chat template (text only)
4 chat_text = tokenizer.apply_chat_template(
5     prompt,
6     add_generation_prompt=True,
7     tokenize=False
8 )
9
10 # Tokenize
11 inputs = tokenizer(
12     chat_text,
13     return_tensors="pt"
14 ).to(device)
15
16 # Generate output
17 with torch.no_grad():
18     outputs = model.generate(
19         input_ids=inputs["input_ids"],
20         attention_mask=inputs["attention_mask"],
21         generation_config=generation_config,
22     )
23
24
25 # Decode output
26 output_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
27
28
29 # print
30 if "<|assistant|>" in output_text:
31     print(output_text.split("<|assistant|>")[-1].strip())
32 else:
33     print(output_text.strip())
```



Instruction Tuning

❖ Step 4: Test pretrained model performance

system

Cutting Knowledge Date: December 2023

Today Date: 26 Jul 2024

You are a helpful AI assistant developed by Meta. Respond safely and accurately.user

Analyze the given passage and question. Choose the best answer from the options below.

Passage:

[This passage is adapted from George Eliot, *Silas Marner*. Originally published in 1861. Silas was

...

Silas puzzled on such occasions by the incompatible demands of love.]

Question:

Which statement best describes a technique the narrator uses to represent Silas's character before he adopted Eppie?

Choices:

- A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold as requiring certain behaviors on his part.
- B) The narrator underscores Silas's former greed by describing his gold as seeming to reproduce on its own.
- C) The narrator hints at Silas's former antisocial attitude by contrasting his present behavior toward his neighbors with his past behavior toward them.
- D) The narrator demonstrates Silas's former lack of self-awareness by implying that he is unable to recall life before Eppie.

Respond ONLY with the letter and full text of the correct answer choice.assistant

A)

Unlike the gold which needed nothing, and must be worshipped in close-locked solitude—which was hidden away from the daylight, was deaf to the song of birds, and started to no human tones—Eppie was a

We will apply fine-tuning to improve pre-trained performance.

Instruction Tuning

❖ Step 5: Download dataset

text	answer	requires_line	id
string · lengths 3.37k 5.63k	string · classes 4 values	bool 2 classes	string · lengths 25 26
SAT READING COMPREHENSION TEST This passage is adapted...	A	false	sat-practice_7-question_3
SAT READING COMPREHENSION TEST This passage is adapted...	D	true	sat-practice_7-question_10
SAT READING COMPREHENSION TEST This passage is adapted...	C	true	sat-practice_7-question_7
SAT READING COMPREHENSION TEST This passage is adapted...	D	false	sat-practice_7-question_1

Fine-tune LLaMA on
[emozilla/sat-reading](#).



Instruction Tuning

❖ Step 5: Download dataset

```
1 data = load_dataset('emozilla/sat-reading')
```

```
1 data

DatasetDict({
    train: Dataset({
        features: ['text', 'answer', 'requires_line', 'id'],
        num_rows: 298
    })
    validation: Dataset({
        features: ['text', 'answer', 'requires_line', 'id'],
        num_rows: 39
    })
    test: Dataset({
        features: ['text', 'answer', 'requires_line', 'id'],
        num_rows: 38
    })
})
```

```
1 type(data)
```

datasets.dataset_dict.DatasetDict

```
1 print(data["train"]['text'][0])
```

SAT READING COMPREHENSION TEST

This passage is adapted from George Eliot, *Silas Marner*. Originally published in 1861. Silas was a weaver and a notorious miser, but then the gold he had hoarded was stolen. Shortly after, Silas adopted a young child, Eppie, the daughter of an impoverished woman who had died suddenly.

Unlike the gold which needed nothing, and must be worshipped in close-locked solitude—which was
...

Silas puzzled on such occasions by the incompatible demands of love.

Question 3:

Which statement best describes a technique the narrator uses to represent Silas's character?
A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold
B) The narrator underscores Silas's former greed by describing his gold as seeming to
C) The narrator hints at Silas's former antisocial attitude by contrasting his present
D) The narrator demonstrates Silas's former lack of self-awareness by implying that he

Answer:

```
1 print(data["train"]["answer"][0])
```

A

Instruction Tuning

❖ Step 5.1: Preprocess dataset

```

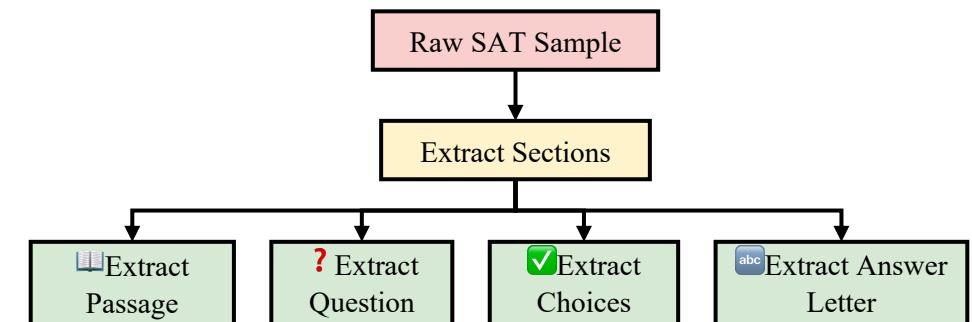
1 import re
2
3 def extract_sections(text):
4     """Parse raw SAT text into structured sections"""
5     sections = {
6         'passage': '',
7         'question': '',
8         'choices': [],
9         'answer_letter': ''
10    }
11
12     answer_part = text.split('Answer:')[-1].strip()
13     sections['answer_letter'] = answer_part[0] if answer_part else ''
14
15     content = text.split('SAT READING COMPREHENSION TEST')[-1].split('Answer:')[0]
16     blocks = [b.strip() for b in content.split('\n\n') if b.strip()]
17
18     passage_lines = []
19     for line in blocks:
20         if line.startswith('Question'):
21             break
22         passage_lines.append(line)
23     sections['passage'] = '\n'.join(passage_lines).strip()
24

```

```

25     for block in blocks:
26         if block.startswith('Question'):
27             lines = block.split('\n')
28             question_lines = []
29             choice_lines = []
30
31             for line in lines[1:]:
32                 if re.match(r'^[A-D]\)', line.strip()):
33                     choice_lines.append(line.strip())
34                 else:
35                     question_lines.append(line.strip())
36
37             sections['question'] = '\n'.join(question_lines).strip()
38             sections['choices'] = choice_lines
39
40     return sections

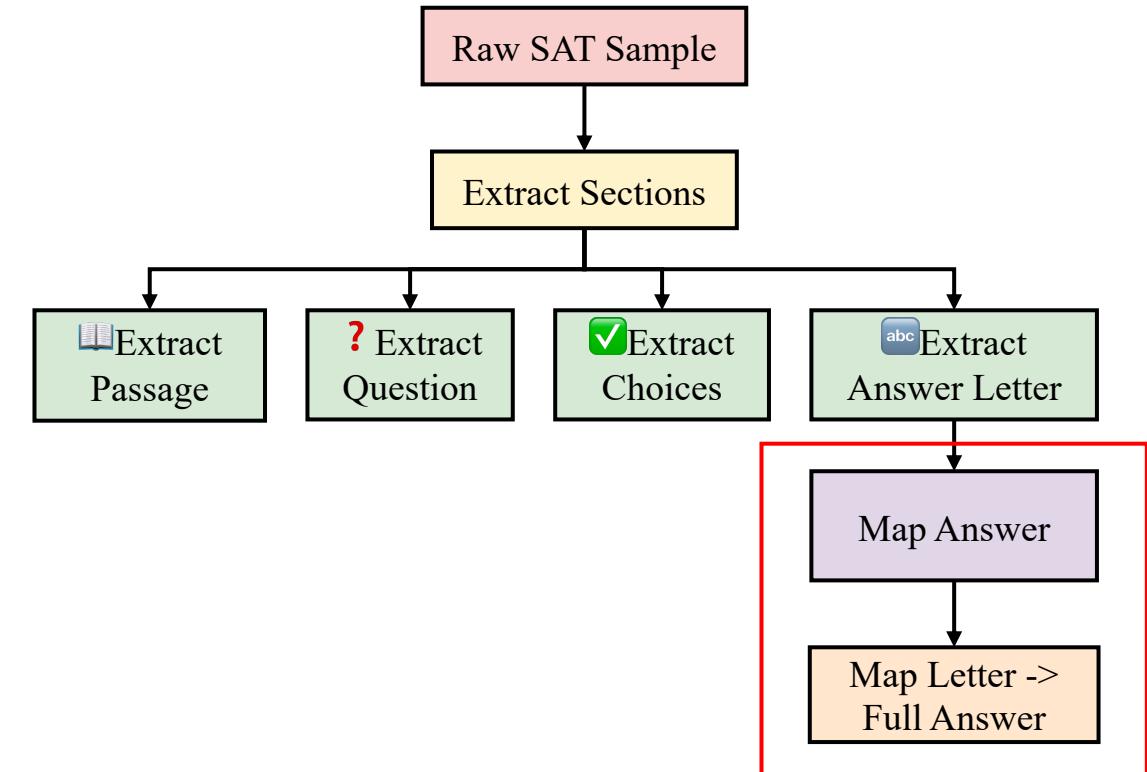
```



Instruction Tuning

❖ Step 5.1: Preprocess dataset

```
1 def map_answer(text, letter):
2     """Match answer letter with full choice text"""
3     sections = extract_sections(text)
4     for choice in sections['choices']:
5         if choice.startswith(f"{{letter}}"):
6             return choice
7     return letter
```





Instruction Tuning

❖ Prompting format

system

Cutting Knowledge Date: December 2023

Today Date: 26 Jul 2024

You are a helpful AI assistant developed by Meta. Respond safely and accurately.user

Analyze the given passage and question. Choose the best answer from the options below.

Passage:

Anna sat by the window every day, watching the rain fall.

She missed the sunshine, but she never said it out loud.

Instead, she smiled at everyone and kept her feelings to herself.

Question:

What does the passage suggest about Anna?

Choices:

- A) She enjoys rainy days.
- B) She hides her true feelings.
- C) She likes to talk about the weather.
- D) She avoids sitting by the window.

Respond ONLY with the letter and full text of the correct answer choice.assistant

- B) She hides her true feelings.



Instruction Tuning

❖ Step 6: Create generate prompt function

```
1 def generate_prompt(text, answer_letter):
2     sections = extract_sections(text)
3
4     choices_text = '\n'.join(sections['choices'])
5
6     return [
7         {
8             "role": "system",
9             "content": LLAMA3_SYSTEM_PROMPT
10        },
11        {
12            "role": "user",
13            "content": f"""Read the passage and answer the question.
14
15 ### Passage:
16 {sections['passage']}
17
18 ### Question:
19 {sections['question']}
20
21 ### Choices:
22 {choices_text}
23
24 Respond with ONLY the letter and full text of the correct answer."""
25        },
26        {
27            "role": "assistant",
28            "content": map_answer(text, answer_letter)
29        }
30    ]
31
```

```
1 def generate_and_tokenize_prompt(user_input, answer):
2     try:
3         full_prompt = generate_prompt(user_input, answer)
4
5         prompt_str = tokenizer.apply_chat_template(
6             full_prompt,
7             tokenize=False,
8             add_generation_prompt=False
9         )
10
11         tokenized = tokenizer(
12             prompt_str,
13             padding="max_length",
14             truncation=True,
15             max_length=1506,
16             return_tensors="pt"
17         )
18
19         input_ids = tokenized["input_ids"][0]
20         labels = input_ids.clone()
21
22         return {
23             "input_ids": input_ids,
24             "attention_mask": tokenized["attention_mask"][0],
25             "labels": labels
26         }
27
28     except Exception as e:
29         print(f"Error processing sample: {e}")
30         return None
```



Instruction Tuning

❖ Step 6: Create generate prompt function

== Generated Prompt ==

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

Cutting Knowledge Date: December 2023

Today Date: 21 Apr 2025

You are a helpful AI assistant developed by Meta. Respond safely and accurately.<|eot_id|><|start_header_id|>user<|end_header_id|>

Read the passage and answer the question.

Passage:

This passage is adapted from George Eliot, *Silas Marner*. Originally published in 1861. Silas was a weaver and a notorious miser, but then the gold he had hoarded was stolen. Shortly after, Silas adopted a young child, Eppie, the daughter of an impoverished woman who had died suddenly.

Unlike the gold which needed nothing, and must be worshipped in close-locked solitude—which was

...

Silas puzzled on such occasions by the incompatible demands of love.

Question:

Which statement best describes a technique the narrator uses to represent Silas's character before he adopted Eppie?

Choices:

- A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold as requiring certain behaviors on his part.
- B) The narrator underscores Silas's former greed by describing his gold as seeming to reproduce on its own.
- C) The narrator hints at Silas's former antisocial attitude by contrasting his present behavior toward his neighbors with his past behavior toward them.
- D) The narrator demonstrates Silas's former lack of self-awareness by implying that he is unable to recall life before Eppie.

Respond with ONLY the letter and full text of the correct answer.<|eot_id|><|start_header_id|>assistant<|end_header_id|>

- A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold as requiring certain behaviors on his part.<|eot_id|>

```
1 sample_text = data["train"]['text'][0]
2 sample_answer = data["train"]['answer'][0]
3
4 example_messages = generate_prompt(sample_text, sample_answer)
5 prompt_text = tokenizer.apply_chat_template(
6     example_messages,
7     tokenize=False,
8     add_generation_prompt=False
9 )
10 print("== Generated Prompt ==")
11 print(prompt_text)
12
```

Instruction Tuning

❖ Step 6: Create generate prompt function

```
== Tokenized Sample ==
Input IDs shape: torch.Size([1506])
Sample decoded back:
system
```

Cutting Knowledge Date: December 2023
Today Date: 21 Apr 2025

You are a helpful AI assistant developed by Meta. Respond safely and accurately.user

Read the passage and answer the question.

Passage:

This passage is adapted from George Eliot, *Silas Marner*. Originally published in 1861. Silas was a weaver and a notorious miser, but then the gold he had hoarded was stolen. Shortly after, Silas adopted a young child, Eppie, the daughter of an impoverished woman who had died suddenly. Unlike the gold which needed nothing, and must be worshipped in close-locked solitude—which was

...

for his watchfulness and penetration. Sorely was poor Silas puzzled on such occasions by the incompatible demands of love.

Question:

Which statement best describes a technique the narrator uses to represent Silas's character before he adopted Eppie?

Choices:

- A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold as requiring certain behaviors on his part.
- B) The narrator underscores Silas's former greed by describing his gold as seeming to reproduce on its own.
- C) The narrator hints at Silas's former antisocial attitude by contrasting his present behavior toward his neighbors with his past behavior toward them.
- D) The narrator demonstrates Silas's former lack of self-awareness by implying that he is unable to recall life before Eppie.

Respond with ONLY the letter and full text of the correct answer.assistant

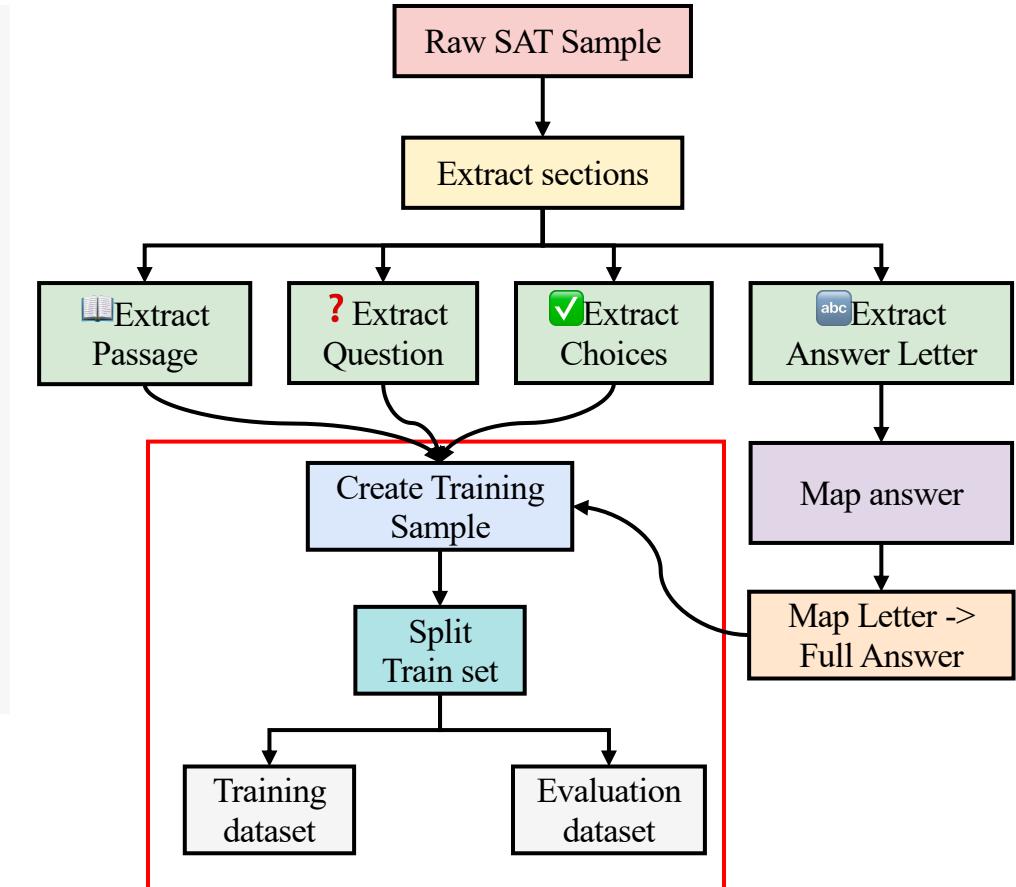
- A) The narrator emphasizes Silas's former obsession with wealth by depicting his gold as requiring certain behaviors on his part.

Instruction Tuning

❖ Step 7: Create training samples

```
1 from sklearn.model_selection import train_test_split
2
3 training_samples = []
4 for sample in tqdm(data['train']):
5     try:
6         # preprocessing data
7         processed_text = sample['text'].replace('SAT READING COMPREHENSION TEST', '').strip()
8         processed_answer = map_answer(sample['text'], sample['answer'].strip())
9
10        # create sample
11        tokenized_sample = generate_and_tokenize_prompt(processed_text, processed_answer)
12        if tokenized_sample is not None:
13            training_samples.append(tokenized_sample)
14    except Exception as e:
15        print(f"Skipping invalid sample: {e}")
16
17 training_samples = [s for s in training_samples if s is not None]
18
19 train_samples, val_samples = train_test_split(training_samples, test_size=0.1, random_state=42)
20 train_dataset = Dataset.from_list(train_samples)
21 eval_dataset = Dataset.from_list(val_samples)
```

100% | 298/298 [00:00<00:00, 335.27it/s]



Instruction Tuning

❖ Step 8: Training

```
1 from transformers import TrainerCallback
2 from rich.console import Console
3 from rich.table import Table
4
5 class LogLossCallback(TrainerCallback):
6     def __init__(self):
7         self.console = Console()
8         self.table = Table(show_header=True, header_style="bold magenta")
9         self.table.add_column("Step", justify="right")
10        self.table.add_column("Training Loss", justify="right")
11        self.logged_steps = set()
12
13    def on_log(self, args, state, control, logs=None, **kwargs):
14        if logs is not None and "loss" in logs:
15            step = state.global_step
16            if step not in self.logged_steps:
17                loss = logs["loss"]
18                self.table.add_row(str(step), f"{loss:.6f}")
19                self.logged_steps.add(step)
20
21            if step % 10 == 0:
22                self.console.print(self.table)
23
```

```
1 from transformers import TrainingArguments, Trainer, DataCollatorForLanguageModeling
2
3 training_args = TrainingArguments(
4     per_device_train_batch_size=1,
5     gradient_accumulation_steps=2,
6     num_train_epochs=2,
7     learning_rate=2e-4,
8     fp16=True,
9     save_total_limit=3,
10    logging_steps=10,
11    output_dir="llama3-8b-sat-reading",
12    optim="paged_adamw_8bit",
13    lr_scheduler_type="cosine",
14    warmup_ratio=0.05,
15    eval_strategy="steps",
16    eval_steps=50,
17    save_strategy="steps",
18    save_steps=50,
19    load_best_model_at_end=True,
20    metric_for_best_model="loss",
21    greater_is_better=False,
22    report_to="none",
23    remove_unused_columns=False
24 )
25
26 data_collator = DataCollatorForLanguageModeling(
27     tokenizer=tokenizer,
28     mlm=False,
29     pad_to_multiple_of=8
30 )
```

Instruction Tuning

❖ Step 8: Training

```
33 trainer = Trainer(  
34     model=model,  
35     train_dataset=train_dataset,  
36     eval_dataset=eval_dataset,  
37     args=training_args,  
38     data_collator=data_collator,  
39     callbacks=[LogLossCallback()]  
40 )  
41  
42 # Quantization-aware training settings  
43 model.config.use_cache = False  
44 model.enable_input_require_grads()  
45 model = torch.compile(model)  
46  
47 trainer.train()
```

[268/268 11:59, Epoch 2/2]

Step	Training Loss	Validation Loss
50	1.778500	1.826047
100	0.957600	1.033670
150	0.335900	0.402805
200	0.228900	0.212004
250	0.157400	0.169596

Step	Training Loss
10	2.602700
20	2.419200
30	2.096300
40	1.735000
50	1.778500
60	1.664100
70	1.642400
80	1.468400
90	1.400300
100	0.957600
110	0.952000
120	0.695000
130	0.673000
140	0.463700
150	0.335900
160	0.343900
170	0.235100
180	0.274100
190	0.241200
200	0.228900
210	0.176200
220	0.169500
230	0.173400
240	0.135600
250	0.157400
260	0.124800



Instruction Tuning

❖ Step 9: Testing & Evaluation

```
1 generation_config = GenerationConfig(  
2     max_new_tokens=64,  
3     temperature=0.0,  
4     top_p=1.0,  
5     do_sample=False,  
6     repetition_penalty=1.0,  
7     eos_token_id=tokenizer.eos_token_id,  
8     pad_token_id=tokenizer.pad_token_id  
9 )  
10  
11 def extract_answer(output_text):  
12     if "<|assistant|>" in output_text:  
13         answer_part = output_text.split("<|assistant|>")[-1].strip()  
14     else:  
15         answer_part = output_text.split("assistant")[-1].strip()  
16  
17     match = re.search(r"^(A-D))\s*(\n(.*))", answer_part, re.MULTILINE)  
18  
19     if match:  
20         return f"{match.group(1)} {match.group(2).strip()}"  
21     else:  
22         clean_lines = [line for line in answer_part.split('\n') if not line.startswith('**')]  
23         return clean_lines[0].strip() if clean_lines else answer_part  
24
```

```
25 def format_test_prompt(text, answer_letter=None):  
26     """Format input text as chat conversation (for prediction or test)"""  
27     sections = extract_sections(text)  
28  
29     # Build choices block  
30     choices_text = '\n'.join(sections['choices'])  
31  
32     user_prompt = f"""Read the passage and answer the question.  
33  
34 ### Passage:  
35 {sections['passage']}  
36  
37 ### Question:  
38 {sections['question']}  
39  
40 ### Choices:  
41 {choices_text}  
42  
43 Respond with ONLY the letter and full text of the correct answer."""  
44  
45     messages = [  
46         {  
47             "role": "system",  
48             "content": LLAMA3_SYSTEM_PROMPT  
49         },  
50         {  
51             "role": "user",  
52             "content": user_prompt  
53         }  
54     ]  
55  
56     if answer_letter:  
57         messages.append({  
58             "role": "assistant",  
59             "content": map_answer(text, answer_letter)  
60         })  
61  
62     return messages
```



Instruction Tuning

❖ Step 9: Testing & Evaluation

```
64 def predict(text):
65     messages = format_test_prompt(text)
66
67     prompt_text = tokenizer.apply_chat_template(
68         messages,
69         add_generation_prompt=True,
70         tokenize=False
71     )
72
73     inputs = tokenizer(prompt_text, return_tensors="pt").to(device)
74
75     with torch.no_grad():
76         outputs = model.generate(
77             input_ids=inputs["input_ids"],
78             attention_mask=inputs["attention_mask"],
79             generation_config=generation_config
80         )
81
82     # Decode
83     output_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
84
85     # === DEBUG ===
86     print("== Full Decoded Output ==")
87     print(output_text)
88
89     # Only answer
90     return extract_answer(output_text)
```

```
1 test_sample_idx = 4
2 input_text = data["test"]['text'][test_sample_idx]
3 true_answer = data["test"]['answer'][test_sample_idx]
4
5 predicted_answer = predict(input_text)
6
7 true_answer_full = map_answer(input_text, true_answer)
8
9 print("\n==== Final Result ===")
10 print(f"[Model Prediction]\n{predicted_answer}")
11 print(f"\n[Ground Truth]\n{true_answer_full}")
```

Question:
In the passage, Threestep is mainly presented as a

Choices:
A) summer retreat for vacationers.
B) small rural town.
C) town that is home to a prominent university.
D) comfortable suburb.

Respond with ONLY the letter and full text of the correct answer.assistant

B) small rural town.

==== Final Result ===

[Model Prediction]

B) small rural town.

[Ground Truth]

B) small rural town.



Instruction Tuning

❖ Step 9: Testing & Evaluation

```
3 def evaluate(test_dataset, max_samples=None):
4     """
5         Evaluate model accuracy on test set
6     Args:
7         test_dataset: Dataset object containing 'text' and 'answer'
8         max_samples: Optional limit for quick testing
9     """
10    correct = 0
11    total = 0
12    wrong_samples = []
13
14    # Process samples with progress bar
15    for idx in tqdm(range(len(test_dataset[:max_samples]['text']))):
16        try:
17            text = test_dataset['text'][idx]
18            true_answer = test_dataset['answer'][idx].strip().upper()
19
20            # Get model prediction
21            predicted = predict(text)
22
23            # Extract first valid choice letter from prediction
24            predicted_letter = re.search(r'\b([A-D])\b', predicted.upper())
25            if predicted_letter:
26                predicted_letter = predicted_letter.group(1)
27            else:
28                predicted_letter = None
29
30            # Compare with ground truth
31            if predicted_letter == true_answer:
32                correct += 1
33            else:
34                wrong_samples.append({
35                    'text': text,
36                    'predicted': predicted,
37                    'true': true_answer
38                })
39
40    total += 1
41
42    except Exception as e:
43        print(f"Error processing sample {idx}: {str(e)}")
44
45    # Calculate metrics
46    accuracy = correct / total if total > 0 else 0
47
48    # Print summary
49    print("\n==== Evaluation Results ===")
50    print(f"Correct: {correct}/{total}")
51    print(f"Accuracy: {accuracy:.2%}")
52    print(f"Wrong samples saved in 'wrong_samples' list")
53
54    return {
55        'accuracy': accuracy,
56        'total': total,
57        'wrong_samples': wrong_samples
58    }
59
60 # Usage
61 test_results = evaluate(data['test'])

==== Evaluation Results ===
Correct: 24/38
Accuracy: 63.16%
Wrong samples saved in 'wrong_samples' list
```

Instruction Tuning

❖ Step 9: Testing & Evaluation

```
1 for i, sample in enumerate(test_results['wrong_samples'][:5]):  
2     print(f"## Sample {i+1}")  
3     print(f"Ground Truth: {sample['true']}")  
4     print(f"Predicted: {sample['predicted']}")  
5     print("Passage:")  
6     print(extract_sections(sample['text'])['passage'][:200] + "...")  
7     print("\n" + "-"*50)
```

```
### Sample 1  
Ground Truth: D  
Predicted: A) sympathy, because they assume that she is experiencing intense heat for the first time.  
Passage:  
This passage is adapted from Mary Helen Stefaniak, The  
Cailiffs of Baghdad, Georgia: A Novel. ©2010 by Mary Helen  
Stefaniak.  
Miss Grace Spivey arrived in Threestep, Georgia,  
in August 1938. She steppe...  
-----  
### Sample 2  
Ground Truth: A  
Predicted: D) Miss Spivey herself.  
Passage:  
This passage is adapted from Mary Helen Stefaniak, The  
Cailiffs of Baghdad, Georgia: A Novel. ©2010 by Mary Helen  
Stefaniak.  
Miss Grace Spivey arrived in Threestep, Georgia,  
in August 1938. She steppe...  
-----  
### Sample 3  
Ground Truth: C  
Predicted: A) delighted.  
Passage:  
This passage is adapted from Mary Helen Stefaniak, The  
Cailiffs of Baghdad, Georgia: A Novel. ©2010 by Mary Helen  
Stefaniak.  
Miss Grace Spivey arrived in Threestep, Georgia,  
in August 1938. She steppe...
```

Instruction Tuning

❖ Step 10: Save model

```
1 model.save_pretrained("trained-model")
```

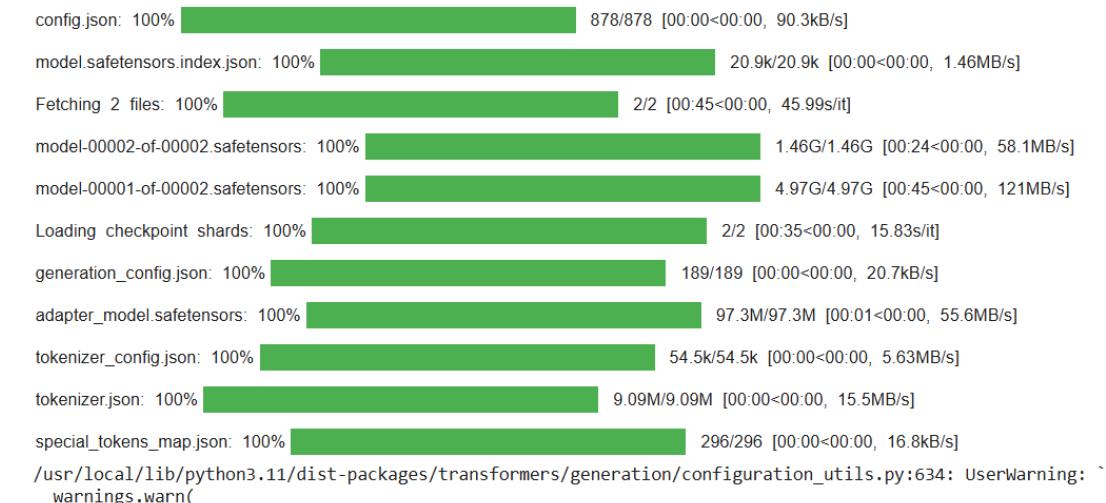
```
1 PEFT_MODEL = "KoiiVN/llama3-3B-peft-SAT-reading-v2"
2
3 model.push_to_hub(
4     PEFT_MODEL, use_auth_token=True
5 )
```

```
adapter_model.safetensors: 100%|██████████| 97.3M/97.3M [00:13<00:00, 7.07MB/s]
CommitInfo(commit_url='https://huggingface.co/KoiiVN/llama3-3B-peft-SAT-reading-v2/commit/c8aef13dae374878afc1f909e955d183a094c9bf', pr_url=None, repo_url=RepoUrl('https://huggingface.co/KoiiVN/llama3-3B-peft-SAT-reading-v2'), pr_revision=None, pr_num=None)
```

Instruction Tuning

❖ Step 11: Inference

```
1 PEFT_MODEL = "KoiVN/llama3-3B-peft-SAT-reading-v2"
2
3 # Load config & model
4 config = PeftConfig.from_pretrained(PEFT_MODEL)
5 bnb_config = BitsAndBytesConfig(
6     load_in_4bit=True,
7     bnb_4bit_use_double_quant=True,
8     bnb_4bit_quant_type="nf4",
9     bnb_4bit_compute_dtype=torch.bfloat16
10 )
11
12 model = AutoModelForCausalLM.from_pretrained(
13     config.base_model_name_or_path,
14     quantization_config=bnb_config,
15     device_map="auto",
16     trust_remote_code=True
17 )
18 model = PeftModel.from_pretrained(model, PEFT_MODEL)
19
20 # Tokenizer & generation config
21 tokenizer = AutoTokenizer.from_pretrained(config.base_model_name_or_path)
22 tokenizer.pad_token = tokenizer.eos_token
```





Instruction Tuning

❖ Step 11: Inference

```
1 def custom_predict(passage: str, question: str, choices: list):
2     choices_text = '\n'.join(choices)
3
4     messages = [
5         {
6             "role": "system",
7             "content": LLAMA3_SYSTEM_PROMPT
8         },
9         {
10            "role": "user",
11            "content": f"""Read the passage and answer the question.
12
13 ### Passage:
14 {passage}
15
16 ### Question:
17 {question}
18
19 ### Choices:
20 {choices_text}
21
22 Respond with ONLY the letter and full text of the correct answer. """
23     ]
24
25
```

```
26     prompt = tokenizer.apply_chat_template(
27         messages,
28         add_generation_prompt=True,
29         tokenize=False
30     )
31
32     inputs = tokenizer(prompt, return_tensors="pt").to(device)
33     with torch.no_grad():
34         outputs = model.generate(
35             input_ids=inputs.input_ids,
36             attention_mask=inputs.attention_mask,
37             generation_config=generation_config
38         )
39
40     full_output = tokenizer.decode(outputs[0], skip_special_tokens=False)
41
42     if "<|assistant|>" in full_output:
43         response = full_output.split("<|assistant|>")[1].replace("<|eot_id|>", "").strip()
44     else:
45         response = full_output.replace(prompt, "").strip()
46
47     return response
```

Instruction Tuning

❖ Step 11: Inference

```
1 custom_passage = """
2 This passage is adapted from F. Scott Fitzgerald, The Great Gatsby.
3 "In my younger and more vulnerable years my father gave me some advice
4 that I've been turning over in my mind ever since. 'Whenever you feel like
5 criticizing anyone,' he told me, 'just remember that all the people in this
6 world haven't had the advantages that you've had.' He didn't say any more,
7 but we've always been unusually communicative in a reserved way, and I
8 understood that he meant a great deal more than that. In consequence, I'm
9 inclined to reserve all judgments, a habit that has opened up many curious
10 natures to me and also made me the victim of not a few veteran bores."
11 """
12
13 custom_question = "What is the primary purpose of the narrator's recollection of his father's advice?"
14 custom_choices = [
15     "A) To explain his reluctance to judge others", #A is correct, just try testing
16     "B) To highlight his privileged upbringing",
17     "C) To criticize his father's moral values",
18     "D) To foreshadow future conflicts in the story"
19 ]
20
21 prediction = custom_predict(custom_passage, custom_question, custom_choices)
22 print("\n== Custom Test Result ==")
23 print("[Prediction]", prediction)
```

```
== Custom Test Result ==
[Prediction] <|begin_of_text|>A) To explain his reluctance to judge others<|eot_id|>
```



AI

AI VIET NAM
@aivietnam.edu.vn

QUIZ

Summarization and Q&A

Summary

In this study session, we have discussed:

- 1. Training type.**
 1. Pre-training.
 2. Supervised Fine-tuning.
- 2. How to make LLMs adapt to a task?**
 1. Prompting (In-context learning).
 1. One-shot learning: Prompting with 1 example.
 2. Few-shot learning: Prompting with more than 1 example.
 3. Chain-of-Thought: Prompting with reasoning.
 2. Instruction Tuning: By supervised learning LLMs with instruction data.
- 3. How to train LLMs on a single (small) GPU?**
 1. Parameter Efficient Fine-Tuning.
- 4. Apply Instruction Tuning for multiple choice SAT English Reading task.**

Summarization and Q&A

