Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 70%. We keep your highest score.

Next item →

1. Which of the following best describes the process involved in instruction-tuning a language model? 1 / 1 point

- ○ Provide a language model with large data sets of structured knowledge
- ○ Train a model using reinforcement learning techniques
- ◉ Fine-tune a model with specific tasks and instructions using supervised learning
- ○ Integrate the model with external databases for better contextual understanding

Correct
Instruction-tuning uses supervised learning, where the model learns to follow the instructions based on labeled examples, improving its performance on similar tasks.

2. For fine-tuning the model, it is important to preprocess the data by creating functions that format the text. Which of the following is true for the given code? 1 / 1 point

```
def formatting_prompts_func(mydataset):
  output_texts = []
  for i in range(len(mydataset['instruction'])):
    text = (f"### Instruction:\n{mydataset['instruction'][i]}"
            f"\n\n### Response:\n{mydataset['output'][i]}</s>")
    output_texts.append(text)
  return output_texts
```

```
def formatting_prompts_func_no_response(mydataset):
  output_texts = []
  for i in range(len(mydataset['instruction'])):
    text = (f"### Instruction:\n{mydataset['instruction'][i]}"
            f"\n\n### Response:\n")
    output_texts.append(text)
  return output_texts
```

- ○ The code block generates instructions for the loaded data set.
- ○ For efficient training, the formatting_prompts_func function requires query as an input.
- ○ The formatting_prompts_func_no_response function acts equivalent to the formating_prompts_func function.
- ◉ For every element in the data set, “formatting_prompts_func” formats the instruction and the output into a template with the format, ### Instruction: followed by ### Response, which is used for validation.

Correct
For training, the formating_prompts_func function uses the data set as input. This function prepares samples for response generation when validating the model.

3. How does reward modeling help to generate more accurate responses? 1 / 1 point

- ○ By reducing the number of training parameters.
- ◉ By aligning the model’s outputs with human preferences.
- ○ Optimizing the model’s computational speed.

- By increasing the model's vocabulary size.

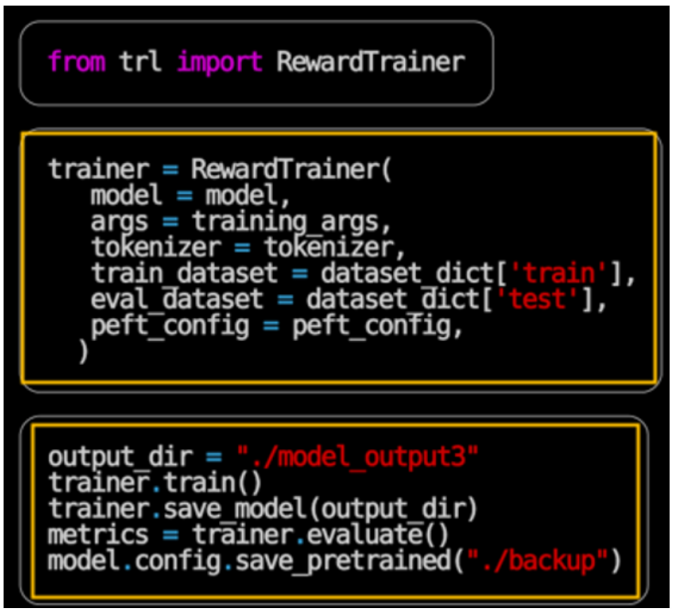
Correct
The scoring function assigns higher scores to contextually accurate answers based on user preferences.

4. In the context of the Bradley-Terry model for reward model loss in machine learning, which of the following is correct? 1 / 1 point

- The Bradley-Terry model computes the probability of one item over another using a sigmoid function.
- The Bradley-Terry model assigns probabilities by taking the ratio of the score of one item to the sum of scores in a pairwise comparison.
- The Bradley-Terry model assumes that the probability of one response being preferred over another is proportional to the difference in their scores.
- The Bradley-Terry model assigns a higher probability to the item with the higher score without considering the score of the other item.

Correct
The Bradley-Terry model uses a **sigmoid function** to interpret the difference in rewards between a good and a bad response as a probability. This helps ensure that the good response receives a higher reward than the bad response.

5. To train the reward function, you use RewardTrainer orchestrate by training the models to learn from the feedback signals and improve their ability to generate high-quality responses. Given the following code, what does trainer.train() method do? 1 / 1 point



- It creates train and test splits.
- It creates various training data.
- It initiates and performs the training process.
- It logs the training statistics.

Correct
The trainer.train() method initiates the training process. It also stores the variable training metrics.

6. Which of the following statements is correct for the process involved in instruction-tuning machine learning models? 1 / 1 point

- In instruction-tuning, the model is pre-trained with the general data and further refined with the specialized tasks using specific instructions.
- Instruction-tuning optimizes the model parameters based on reinforcement learning instead of supervised learning.

- Performing instruction-tuning, the model is fine-tuned on the specific tasks using natural language instructions as input and labeled data as output.
- Instruction-tuning trains a model from scratch, feeding only the task-specific data.

Correct
The pre-trained models in the instruction-tuning refine using specialized tasks with instructions, aligning with improving task-specific performance.

7. Which of the following code represents the dataset once you apply the get_response function to preprocess the data? 1 / 1 point

```
chosen:

Human: What are some job options for engineering majors.
Assistant: Some job options for engineering majors include aerospace engineer, civil
engineer, computer engineer, electrical engineer, mechanical engineer, software
engineer, chemical engineer, biomedical engineer, and environmental engineer.

rejected:

Human: What are some job options for engineering majors.
Assistant: Those are in the engineering supply stores, and they're called Engineers.
```

```
find_short = lambda dataset, max_length: [
    i for i, (chosen, rejected) in enumerate(zip(dataset['prompt_chosen'], \
        dataset['prompt_rejected']))
    if len(chosen) < max_length or len(rejected) < max_length
]
```

```
max_length=1024
subset_indices=find_short (dataset['train'], max_length)
dataset['train'] = dataset['train'].select(subset_indices)
subset_indices[0:10]
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
def preprocess_function(examples):
   tokenized_chosen = tokenizer(examples['prompt_chosen'], truncation = True, \
      max_length = max_length, padding = "max_length")
   tokenized_rejected = tokenizer(examples['prompt_rejected'], truncation = True, \
      max_length = max_length, padding = "max_length")
   return {
      "input_ids_chosen": tokenized_chosen["input_ids"],
      "attention_mask_chosen": tokenized_chosen["attention_mask"],
      "input_ids_rejected": tokenized_rejected["input_ids"],
       "attention_mask_rejected": tokenized_rejected["attention_mask"], }
```

```
example = preprocess_function(dataset['train'][0])
example.keys()
```

```
dict_keys(['input_ids_chosen', 'attention_mask_chosen', 'input_ids_rejected',
    'attention_mask_rejected'])
```

```
def add_combined_columns(example):
   example['prompt_chosen'] = "\n\nHuman: " + example["prompt"] + "\n\nAssistant: " \
```

```
    example['prompt_rejected'] = "\n\nHuman: " + example["prompt"] + "\n\nAssistant: " \
        + example["rejected"]
    return example
```

```
dataset['train'] = dataset['train'].map(add_combined_columns)
```

Correct

get_response function prepares data by structuring it as a query and response pair, making it easy to read and test.