A Universal API: Language Modeling to solve many tasks

The Language Modeling objective seems simple, but appearances are deceiving.

We have shown that a Language Model can be adapted for new Target tasks

- via the Unsupervised Pre-Training + Supervised Fine-Tuning paradigm.

There is, perhaps, an alternative way to adapt to a new Target task

- turn each task *directly* into an instance of the Language Modeling objective.

Consider a new Target task: Entailment

- Input is a *pair* of text sequences $[\text{Premise}, \text{Hypothesis}]$
- Binary classification: Does the Hypothesis Logically follow from the Premise ?

```
Premise: The students are attending a lecture on Machine Learning
Hypothesis: The students are sitting in a class room
Label: Entails
```

Suppose we construct a sequence for the Language Model to extend

- consisting of the Premise, Hypothesis, and the string "Label:"

We call this sequence the *prompt* or the *context*

```
Premise: The students are attending a lecture on Machine Learning
Hypothesis: The students are sitting in a class room
Label:
```

We would hope that the Language Model extends the prompt by creating tokens that are the correct response for the Target task

```
Entail
```

since the Hypothesis logically follows from the Premise (other possible response: `Not Entail`)

We have turned Entailment into an instance of Language Modeling.

As another example: consider the Target task: Multiple Choice Question answering

- Input is
    - Context: a sentence or paragraph stating facts
    - Question
    - Answers: a set of possible answer sentences

We construct a prompt of the form

```
Context: It is December of 2022.  Prof. Perry is teaching the second half of th
e Machine Learning Course.
Question: Where are the students ?
Answer 1: The beach
Answer 2: In a classroom in Brooklyn
Answer 3: Dreaming of being elsewhere.

Label:
```

and hope that the Language Model extends the prompt string with one of

```
{ Answer 1, Answer 2, Answer 3 }
```

- hopefully with probability near 100% for Answer 2 !

These transformations of other tasks into instances of the Language Modeling task

- show that the input/output behavior of a Language Model
    - sequence to sequence; specifically: text to text
- enables a Universal API where all tasks can be transformed into completion of a prompt.

The trick is to condition the Language Model to complete the prompt in a manner that solves the Target task.

We would ordinarily do this by

- constructing labeled training examples that demonstrate the Target task :
$$\langle \mathbf{x^{(i)}}, \mathbf{y^{(i)}} \rangle = \langle \mathrm{prompt}, \mathrm{response} \rangle$$
- Fine-Tuning the Pre-Trained model on these demonstration examples

In-Context Learning

Using the Universal API

- we can condition the Pre-Trained Language Model
- to complete the prompt of a new Target Task
- with the correct response for the Target Task
- **without** further training (Fine-Tuning)
- merely by providing the examples that demonsrate the behavior of the new Target task
- **as parts of the prompt**

This is called *In-Context Learning*

- "In-Context" because the demonstration examples are part of the inference-time prompt
    - **not** examples for further training via Fine-Tuning
- The demonstration examples are called *exemplars*

Thus, In-Context Learning constructs a prompt

- with a *pre-prompt*: some number k of exemplars: $\langle \mathrm{prompt}, \mathrm{response} \rangle$ pairs
- followed by an example with no `response`
    - only the example's `prompt`
- with the expectation that the Language model will complete the prompt with the response for the last example.

In a subsequent module, we will explore In-Context Learning in more detail.

In [2]:
```
print("Done")
```

```
Done
```