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 !

Here is what a Language Translation task would look like in text-completion form:

Task: English to French

Context:	English: Please unscramble the letters in the word and write that word
	French:
Target completion:	Veuillez déchiffrer les lettres du mot et écrire ce mot

- Translation task encoded as "predict the next" words following the "French:" prompt.

Power of text as the Universal API

The Universal API turns each task into an instance of the Language Modeling task.

What this means is that

- we might not need *task-specific* models
- just transform every task into instance of the "predict the next" word task

After converting our Source task into a LLM task

- we would ordinarily use Supervised Fine-Tuning
- on a dataset of (transformed) examples from the Target Task
- to Fine-Tune the LLM for the Target task

The transformed Target task training dataset would consist of examples like

$$\langle \mathbf{x}^{(\mathbf{i})}, \mathbf{y}^{(\mathbf{i})} \rangle = \langle \text{prompt}, \text{response} \rangle$$

That is

- the "features" turned into text ("prompt")
- the label ("response") is the continuation of the text

Transforming tasks into Text Completion form

Appendix D of the T5 paper (https://arxiv.org/pdf/1910.10683.pdf#page=47) gives examples of how the input text for a given task is transformed into a sequence of words.

Summary

- Structured input is flattened
    - Inputs consisting of multiple sentences (each with a different role)
    - are flattened into a single sequence
    - separated by tags indicating the role of following sentence
- A "task description" is prepended to the input
    - indicating the task to which the example belongs
    - remember: this is multi-task training

Using an LLM as a Universal Model without Text Completion

There are alternatives to turning every task into Text Completion

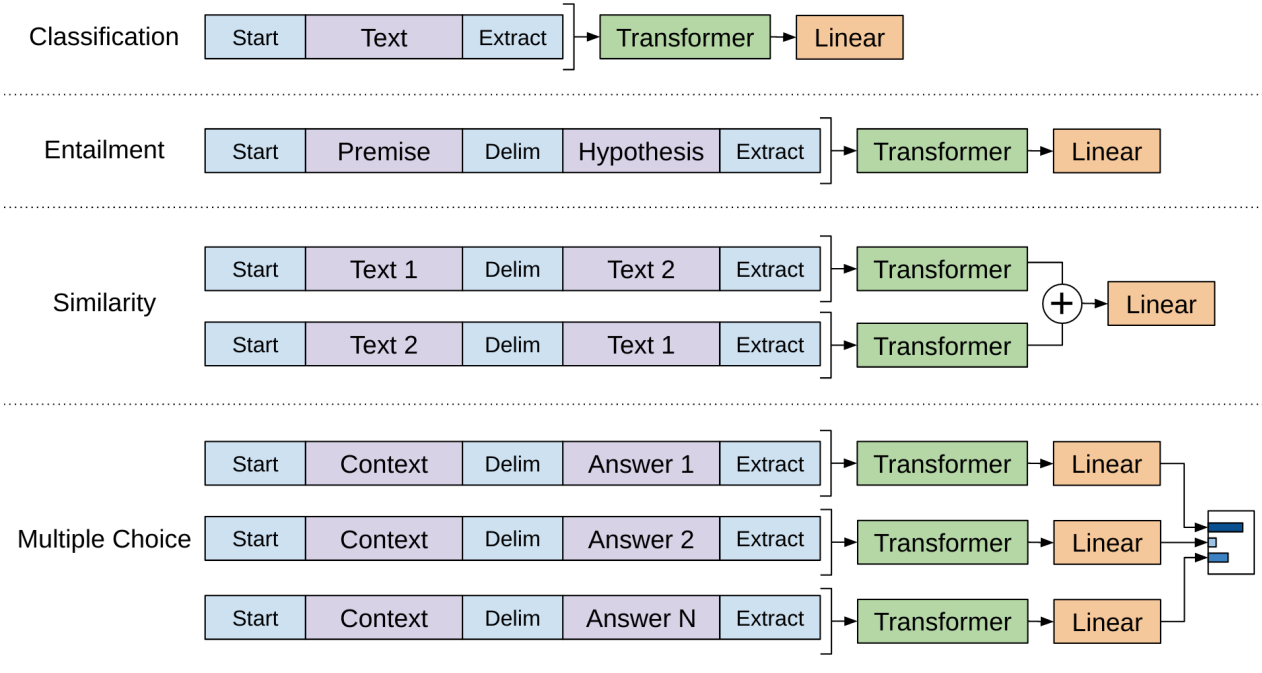
- Pre-processing the example
- Using an LLM
- Post-processing the output

This is certainly not as convenient, but we present it in the spirit of completeness.

Here is a diagram (from the GPT paper) explaining how the authors

- translate the input for the Source task
- into Text input for the multi-task model

Task encoding (from GPT paper)



Picture from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

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

```
Done
```