Steps to a Universal Model for NLP

The fact that the "Unsupervised Pre-Trained Model + Supervised Fine-Tuning" paradigm is so successful in solving new tasks

- suggests that Large Language Models seem to learn a "universal" task-independent representation of language

Still, many tasks don't naturally fit into a Language Model objective.

In this module, we explore the possibility of a Universal Model for all NLP tasks.

The Language Model as a Universal Model (preliminary)

One approach is to convert a Source Task into one or more instances of the Target Task (Language Model).

You may need to

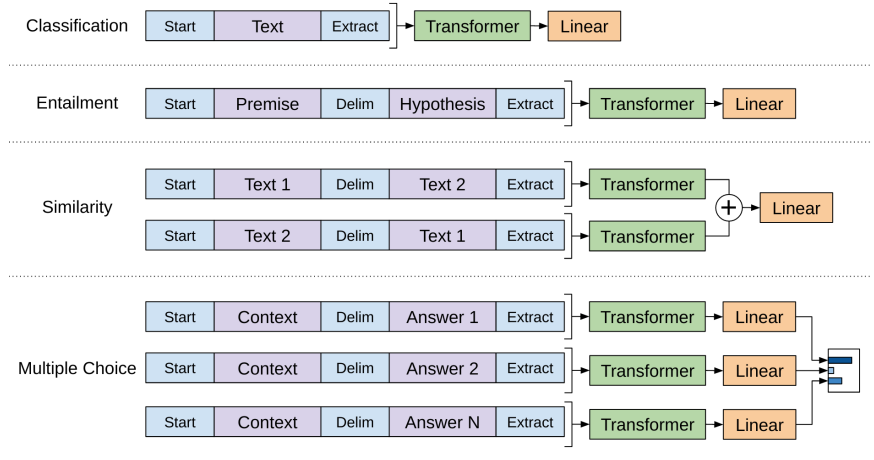
- *transform* your task input from the domain of the Source task
    - into a text string: the domain of the Language Modeling task
- manipulate the output of the Language Model (text) to transform it back into the range of the Target task

We will refer to this step as *Input/Output Transformation*.

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

For example: the Multiple Choice task:

A Universal API: Text to text

The Input/Output transformations get us close to a Universal Model, but not quite

- multiple instances may be required (see Multiple Choice)
- Target-task specific Output transformtion

The next step: create a Universal API: *Text to Text*.

Consider each task as a mapping from Input to Target.

The *Text to Text API*

- converts the Input to Text
- converts the Output to Text

Once both are in Text form

- we combine the texts
- format it into an instance of "Predict the next" (Text completion)
    - the part before the Output is called the *Context* or the *Prompt*
    - the Target Text are the words that "complete" the Input Text ("context")

With the Universal Text To Text API all NLP tasks become

- simple instances of the Language Modeling task
- functions mapping an Input (the "Context') to a Target (the text continuation)

For example, the "unscramble the letters" task becomes

Task: Unscramble the letters

Context:	Please unscramble the letters in the word and write that word
	skicts =
Target completion:	sticks

- The "Unscramble the letters task" encoded as "predict the next" word following the "=" sign

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.

Text to Text

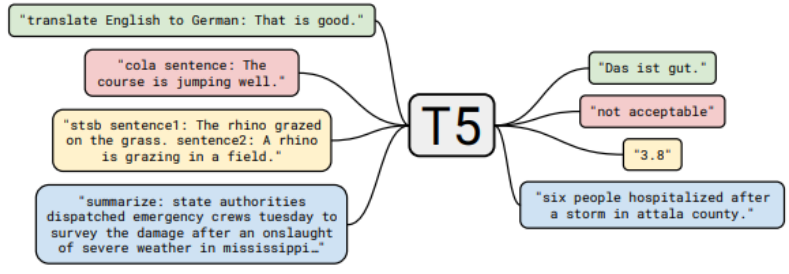


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Attribution: https://arxiv.org/pdf/1910.10683.pdf#page=3

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

The Language Model as the Universal Model

Using the Text to Text API allows us to turn all tasks into instances of a Language Model task.

Note that the Text to Text API produces somewhat "unnatural" text

- Artificial
- May differ from the format of the examples used for Pre-Training

Thus, we should expect to have to Fine-Tune the Language Model on the formated version of examples from the Source Task.

We shall see that it also allows us to

- Present examples of a new Target Task at Inference time (rather than Training time)
- Explore whether a Pre-Trained model **without** Fine-Tuning is able to adapt to the new Target Task
    - **without** adjusting its parameters !
- This is called *In-Context Learning* and will be the subject of a subsequent module

In [1]:
```
print("Done")
```

```
Done
```