Text to Text: A Universal API for NLP

T5: Text to Text Transfer Transformer (https://arxiv.org/pdf/1910.10683.pdf)

T5 blog (https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html)

Colab: Fine-Tuning T5 for Closed Book Answering (https://colab.research.google.com/github/google-research/text-to-text-transfer-transformer/blob/master/notebooks/t5-trivia.ipynb)

- uses GCS and other tools: a little complex

Natural Language Processing is a very broad area, encompassing many seemingly diverse tasks

- Classification: sentiment analysis
- Summarization: short summary of long text
- Translation: from one human language to another
    - Less obvious
        - description of a web page to HTML
        - text to image
- Multiple choice question answering
    - Open book: (https://ai.google.com/research/NaturalQuestions/visualization) paragraph (the "context") is given and the answer is contained within it
        - as a continuous span of tokens
    - Closed book: no context; all "knowledge" obtained from pre-training
        - T5 trivia contest demo (https://t5-trivia.glitch.me/)
- Entailment: does one sentence (the "hypothesis") follow from another (the "premise")

In the early days of Deep Learning, separate models where created (from scratch) for each task.

Transfer Learning is the process of adapting a pre-trained model for a *Source* task into a model for a different *Target* Task.

The process is

- someone *pre-trains* a model for the Source Task on large, general training datasets
- everyone else *fine-tunes* the model on smaller, Target-task specific training datasets.

The more recent approach to NLP in Deep Learning is based on Transfer Learning.

- the representation of text learned for the Source Task
- seems to be useful for many potential Target Tasks

One impediment is that, on the surface, all the NLP tasks seem different.

If we could

- translate each task into an instance of a generic task
- we could have one model for many tasks

In other words: a *Universal API for NLP*.

There are a number of approaches to the Universal API but they can mostly be characterized as *Text to Text*

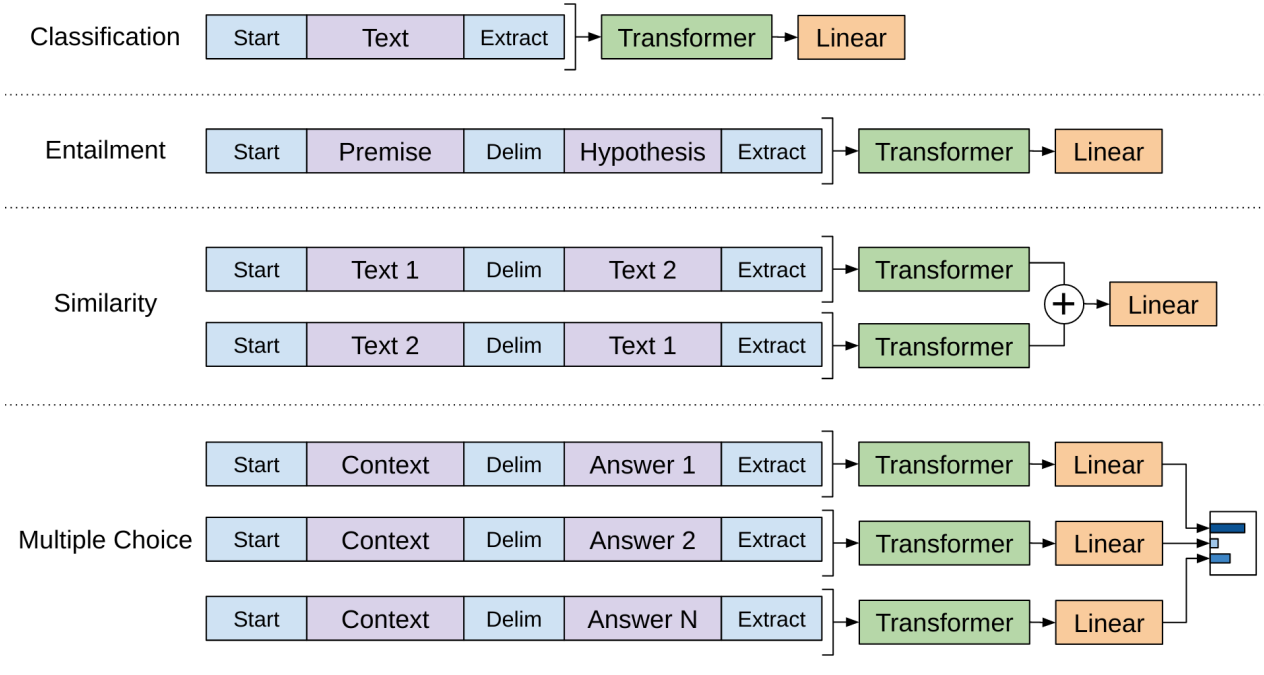
- convert the (potentially structured) input for a particular NLP task to sequence of tokens
- feed to the multi-task model
- the answer to the particular NLP task is the Text output by the model

Text to Text is the Universal API for NLP.

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:

- raw input is structured
    - Context Text
    - N possible answer Texts
- Each potential Answer text
- is concatenated to the Context Text
- the representation of the final `[CLS]` toke is an alternate representation of the Context/Answer pair
- which is fed into a Transformer + Linear network
    - the Linear part just changes the dimensions (matrix multiplication, no activation)
- and softmax'ed to determine which Context/Answer pair is most likely

A quote from the T5: Text to Text Transfer Transformer (https://arxiv.org/pdf/1910.10683.pdf) paper

> *With T5, we propose reframing all NLP tasks into a unified text-to-text-format where the input and output are always text strings, in contrast to BERT-style models that can only output either a class label or a span of the input.*
>
> ***Our text-to-text framework allows us to use the same model, loss function, and hyperparameters on any NLP task, including machine translation, document summarization, question answering, and classification tasks (e.g., sentiment analysis). We can even apply T5 to regression tasks by training it to predict the string representation of a number instead of the number itself.***

T5 Pre-processing

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

Here is an example of

- the Original Input and Target
- processed Input and Target

for an entailment task (MNLI: Multi Language Natural Inference)

Original input:

```
*Hypothesis:* The St. Louis Cardinals have always won.

*Premise:* yeah well losing is i mean i’m i’m originally from Saint Louis and S
aint Louis Cardinals when they were there were uh a mostly a losing team but
```

Processed input:

```
*mnli* *hypothesis:* The St. Louis Cardinals have always won. *premise:* yeah w
ell losing is i mean i’m i’m originally from Saint Louis and Saint Louis Cardin
als when they were there were uh a mostly a losing team but
```

Original target: 2

Processed target: contradiction

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

```
Done
```