# Assignment 3, Part 2: T5 SQuAD Model

Welcome to the part 2 of testing the models for this week's assignment. This time we will perform decoding using the T5 SQuAD model. In this notebook we'll perform Question Answering by providing a "Question", its "Context" and see how well we get the "Target" answer.

## IMPORTANT

- As you cannot save the changes you make to this colab, you have to make a copy of this notebook in your own drive and run that. You can do so by going to `File -> Save a copy in Drive`. Close this colab and open the copy which you have made in your own drive.
- Go to this [google drive folder (https://drive.google.com/drive/folders/1rOZsbEzcpMRVvgrRULRh1JPFpkIG_JOz?usp=sharing)](https://drive.google.com/drive/folders/1rOZsbEzcpMRVvgrRULRh1JPFpkIG_JOz?usp=sharing) named `NLP C4 W3 Colabs & Data`. In the folder, next to its name use the drop down menu to select `"Add shortcut to Drive" -> "My Drive" and then press ADD SHORTCUT`. This should add a shortcut to the folder `NLP C4 W3 Colabs & Data` within your own google drive. Please make sure this happens, as you'll be reading the data for this notebook from this folder.
- Make sure your runtime is GPU (*not* CPU or TPU). And if it is an option, make sure you are using *Python 3*. You can select these settings by going to `Runtime -> Change runtime type -> Select the above mentioned settings and then press SAVE`

**Note: Restarting the runtime maybe required**.

Colab will tell you if the restarting is necessary -- you can do this from the:

Runtime > Restart Runtime

option in the dropdown.

## Outline

# Part 0: Downloading and loading dependencies

Uncomment the code cell below and run it to download some dependencies that you will need. You need to download them once every time you open the colab. You can ignore the `kfac` error.

In [ ]:

```
!pip -q install trax==1.3.4
```

```
     |████████████████████████████████| 368kB 9.7MB/s
     |████████████████████████████████| 163kB 43.7MB/s
     |████████████████████████████████| 1.5MB 51.8MB/s
     |████████████████████████████████| 2.6MB 55.0MB/s
     |████████████████████████████████| 1.1MB 52.1MB/s
     |████████████████████████████████| 71kB 10.8MB/s
     |████████████████████████████████| 3.5MB 54.0MB/s
     |████████████████████████████████| 1.0MB 55.1MB/s
     |████████████████████████████████| 307kB 63.6MB/s
     |████████████████████████████████| 358kB 63.7MB/s
     |████████████████████████████████| 368kB 54.9MB/s
     |████████████████████████████████| 655kB 55.1MB/s
     |████████████████████████████████| 81kB 12.6MB/s
     |████████████████████████████████| 194kB 67.7MB/s
     |████████████████████████████████| 983kB 54.2MB/s
     |████████████████████████████████| 5.3MB 59.6MB/s
     |████████████████████████████████| 890kB 52.8MB/s
     |████████████████████████████████| 3.0MB 51.4MB/s
     |████████████████████████████████| 51kB 8.8MB/s
     |████████████████████████████████| 235kB 57.6MB/s
  Building wheel for bz2file (setup.py) ... done
  Building wheel for pypng (setup.py) ... done
  Building wheel for sacremoses (setup.py) ... done
ERROR: kfac 0.2.2 has requirement tensorflow-probability==0.8, but you'll have tensorflow-probability 0.7.0 which is i
ncompatible.
```

In [12]:

```
import string
import t5
import numpy as np
import trax
from trax.supervised import decoding
import textwrap
# Will come handy later.
wrapper = textwrap.TextWrapper(width=70)
```

# Part 1: Mounting your drive for data accessibility

Run the code cell below and follow the instructions to mount your drive. The data is the same as used in the coursera version of the assignment

In [13]:
```
from google.colab import drive
drive.mount('/content/drive/', force_remount=True)
```

```
Mounted at /content/drive/
```

# Part 2: Getting things ready

Run the code cell below to ready some functions which will later help us in decoding. The code and the functions are the same as the ones you previsouly ran on the coursera version of the assignment.

In [14]:
```python
PAD, EOS, UNK = 0, 1, 2

def detokenize(np_array):
  return trax.data.detokenize(
      np_array,
      vocab_type='sentencepiece',
      vocab_file='sentencepiece.model',
      vocab_dir='/content/drive/My Drive/NLP C4 W3 Data/')

def tokenize(s):
  # The trax.data.tokenize function operates on streams,
  # that's why we have to create 1-element stream with iter
  # and later retrieve the result with next.
  return next(trax.data.tokenize(
      iter([s]),
      vocab_type='sentencepiece',
      vocab_file='sentencepiece.model',
      vocab_dir='/content/drive/My Drive/NLP C4 W3 Data/'))

vocab_size = trax.data.vocab_size(
    vocab_type='sentencepiece',
    vocab_file='sentencepiece.model',
    vocab_dir='/content/drive/My Drive/NLP C4 W3 Data/')

def get_sentinels(vocab_size):
    sentinels = {}

    for i, char in enumerate(reversed(string.ascii_letters), 1):

        decoded_text = detokenize([vocab_size - i])

        # Sentinels, ex: <Z> - <a>
        sentinels[decoded_text] = f'<{char}>'

    return sentinels

sentinels = get_sentinels(vocab_size)


def pretty_decode(encoded_str_list, sentinels=sentinels):
    # If already a string, just do the replacements.
    if isinstance(encoded_str_list, (str, bytes)):
        for token, char in sentinels.items():
            encoded_str_list = encoded_str_list.replace(token, char)
        return encoded_str_list
```

```
    # We need to decode and then prettyfy it.
    return pretty_decode(detokenize(encoded_str_list))
```

# Part 3: Fine-tuning on SQuAD

Now let's try to fine tune on SQuAD and see what becomes of the model. For this, we need to write a function that will create and process the SQuAD `tf.data.Dataset` . Below is how T5 pre-processes SQuAD dataset as a text2text example. Before we jump in, we will have to first load in the data.

### 3.1 Loading in the data and preprocessing

You first start by loading in the dataset. The text2text example for a SQuAD example looks like:

```
{
  'inputs': 'question: <question> context: <article>',
  'targets': '<answer_0>',
}
```

The squad pre-processing function takes in the dataset and processes it using the sentencePiece vocabulary you have seen above. It generates the features from the vocab and encodes the string features. It takes on question, context, and answer, and returns "question: Q context: C" as input and "A" as target.

In [15]:
```
# Retrieve Question, C, A and return "question: Q context: C" as input and "A" as target.
def squad_preprocess_fn(dataset, mode='train'):
  return t5.data.preprocessors.squad(dataset)
```

In [16]:
```python
# train generator, this takes about 1 minute
train_generator_fn, eval_generator_fn = trax.data.tf_inputs.data_streams(
  'squad/v1.1:2.0.0',
  data_dir='/content/drive/My Drive/NLP C4 W3 Data/data/',
  bare_preprocess_fn=squad_preprocess_fn,
  input_name='inputs',
  target_name='targets'
)
train_generator = train_generator_fn()
next(train_generator)
```

Out[16]:
```
(b'question: What year did Harper Lee \' s father represent two black men accused of murder ? context: Lee has said th
at To Kill a Mockingbird is not an autobiography , but rather an example of how an author " should write about what he
knows and write truthfully " . Nevertheless , several people and events from Lee \' s childhood parallel those of the
fictional Scout . Lee \' s father , Amasa Coleman Lee , was an attorney , similar to Atticus Finch , and in 1919 , he
defended two black men accused of murder . After they were convicted , hanged and mutilated , he never tried another c
riminal case . Lee \' s father was also the editor and publisher of the Monroeville newspaper . Although more of a pro
ponent of racial segregation than Atticus , he gradually became more liberal in his later years . Though Scout \' s mo
ther died when she was a baby , Lee was 25 when her mother , Frances Cunningham Finch , died . Lee \' s mother was pro
ne to a nervous condition that rendered her mentally and emotionally absent . Lee had a brother named Edwin , who \xe2
\x80\x94 like the fictional Jem \xe2\x80\x94 was four years older than his sister . As in the novel , a black housekee
per came daily to care for the Lee house and family . ',
 b'1919')
```

In [17]:
```python
#print example from train_generator
(inp, out) = next(train_generator)
print(inp.decode('utf8').split('context:')[0])
print()
print('context:', inp.decode('utf8').split('context:')[1])
print()
print('target:', out.decode('utf8'))
```

```
question: The lower case letters caused a differ in the patter , what did this cause ?

context:  With the other special characters and control codes filled in , ASCII was published as ASA X3 . 4 - 1963 , l
eaving 28 code positions without any assigned meaning , reserved for future standardization , and one unassigned contr
ol code . : 66 , 245 There was some debate at the time whether there should be more control characters rather than the
lowercase alphabet . : 435 The indecision did not last long : during May 1963 the CCITT Working Party on the New Teleg
raph Alphabet proposed to assign lowercase characters to columns 6 and 7 , and International Organization for Standard
ization TC 97 SC 2 voted during October to incorporate the change into its draft standard . The X3 . 2 . 4 task group
voted its approval for the change to ASCII at its May 1963 meeting . Locating the lowercase letters in columns 6 and 7
caused the characters to differ in bit pattern from the upper case by a single bit , which simplified case - insensiti
ve character matching and the construction of keyboards and printers .

target: simplified case - insensitive character matching and the construction of keyboards and printers
```

## 3.2 Decoding from a fine-tuned model

In [18]:
```
# Initialize the model
model = trax.models.Transformer(
    d_ff = 4096,
    d_model = 1024,
    max_len = 2048,
    n_heads = 16,
    dropout = 0.1,
    input_vocab_size = 32000,
    n_encoder_layers = 24,
    n_decoder_layers = 24,
    mode='predict')  # Change to 'eval' for slow decoding.
```

In [19]:
```
# load in the model
# this will take a minute
shape11 = trax.shapes.ShapeDtype((1, 1), dtype=np.int32)
model.init_from_file('/content/drive/My Drive/NLP C4 W3 Data/models/model_squad.pkl.gz',
                     weights_only=True, input_signature=(shape11, shape11))
```

In [ ]:
```
# Uncomment to see the transformer's structure.
print(model)
```

In [21]:
```
# create inputs
# a simple example
# inputs = 'question: She asked him where is john? context: John was at the game'

# an extensive example
inputs = 'question: What are some of the colours of a rose? context: A rose is a woody perennial flowering plant of the
genus Rosa, in the family Rosaceae, or the flower it bears.There are over three hundred species and tens of thousands o
f cultivars. They form a group of plants that can be erect shrubs, climbing, or trailing, with stems that are often arm
ed with sharp prickles. Flowers vary in size and shape and are usually large and showy, in colours ranging from white t
hrough yellows and reds. Most species are native to Asia, with smaller numbers native to Europe, North America, and nor
thwestern Africa. Species, cultivars and hybrids are all widely grown for their beauty and often are fragrant.'
```

In [22]:
```python
# tokenizing the input so we could feed it for decoding
print(tokenize(inputs))
test_inputs = tokenize(inputs)
```

```
[  822    10   363    33   128    13     8  6548    13     3     9  4659
    58  2625    10    71  4659    19     3     9  1679    63 24999  5624
    53  1475    13     8     3   729   302 15641     6    16     8   384
 15641  8433    15     6    42     8  5624    34  4595     7     5  7238
    33   147   386  6189  3244    11     3   324     7    13  2909    13
 10357   291     7     5   328   607     3     9   563    13  2677    24
    54    36     3    15 12621 21675     7     6 11908     6    42  5032
    53     6    28  6269     7    24    33   557     3  8715    28  4816
     3  2246 19376     7     5 20294  5215    16   812    11  2346    11
    33  1086   508    11   504    63     6    16  6548     3  6836    45
   872   190  4459     7    11  1131     7     5  1377  3244    33  4262
    12  3826     6    28  2755  2302  4262    12  1740     6  1117  1371
     6    11  3457 24411  2648     5     3  7727   725     6 10357   291
     7    11  9279     7    33    66  5456  4503    21    70  2790    11
   557    33 29346     5]
```

In [23]:
```python
# Temperature is a parameter for sampling.
#   # * 0.0: same as argmax, always pick the most probable token
#   # * 1.0: sampling from the distribution (can sometimes say random things)
#   # * values inbetween can trade off diversity and quality, try it out!
output = decoding.autoregressive_sample(model, inputs=np.array(test_inputs)[None, :],
                                        temperature=0.0, max_length=10)
print(wrapper.fill(pretty_decode(output[0])))
```

```
white through yellows and reds
```

**Note: As you can see the RAM is almost full, it is because the model and the decoding is memory heavy. You can run decoding just once. Running it the second time with another example might give you the same answer as before, or not run at all (crash). If that happens restart the runtime (see how to at the start of the notebook) and run all the cells again.**