Assignment 3 Ungraded Sections - 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.

Colab

Since this ungraded lab takes a lot of time to run on coursera, as an alternative we have a colab prepared for you.

T5 SQuAD Model Colab

- If you run into a page that looks similar to the one below, with the option `Open with`, this would mean you need to download the `Colaboratory` app. You can do so by `Open with -> Connect more apps -> in the search bar write "Colaboratory" -> install`

- After installation it should look like this. Click on `Open with Google Colaboratory`

Outline

Overview

In this notebook you will:

- Implement the Bidirectional Encoder Representation from Transformer (BERT) loss.
- Use a pretrained version of the model you created in the assignment for inference.

Part 1: Getting ready

Run the code cells below to import the necessary libraries and to define some functions which will be useful for decoding. The code and the functions are the same as the ones you previsouly ran on the graded assignment.

In [1]:

```
import string
import t5
import numpy as np
import trax
from trax.supervised import decoding
import textwrap

wrapper = textwrap.TextWrapper(width=70)
```

```
INFO:tensorflow:tokens_length=568 inputs_length=512 targets_length=114 noise_density=0.15
mean_noise_span_length=3.0
```

In [2]:

```
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='.')


def tokenize(s):
    return next(trax.data.tokenize(
        iter([s]),
        vocab_type='sentencepiece',
        vocab_file='sentencepiece.model',
        vocab_dir='.'))


vocab_size = trax.data.vocab_size(
    vocab_type='sentencepiece',
    vocab_file='sentencepiece.model',
    vocab_dir='.')


def get_sentinels(vocab_size, display=False):
    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}>'
        if display:
            print(f'The sentinel is <{char}> and the decoded token is:', decoded_text)
    return sentinels


sentinels = get_sentinels(vocab_size, display=False)


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 2: 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.

2.1 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 [3]:

```
# 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 [4]:

```
# train generator, this takes about 1 minute
train_generator_fn, eval_generator_fn = trax.data.tf_inputs.data_streams(
  'squad/plain_text:1.0.0',
  data_dir='data/',
  bare_preprocess_fn=squad_preprocess_fn,
  input_name='inputs',
  target_name='targets'
)

train_generator = train_generator_fn()
next(train_generator)
```

Out[4]:

```
(b'question: Which seabird has the longest - distance migration ? context: Seabird migration is si
milar in pattern to those of the waders and waterfowl . Some , such as the black guillemot Cepphus
grylle and some gulls , are quite sedentary ; others , such as most terns and auks breeding in the
temperate northern hemisphere , move varying distances south in the northern winter . The Arctic t
ern Sterna paradisaea has the longest - distance migration of any bird , and sees more daylight th
an any other , moving from its Arctic breeding grounds to the Antarctic non - breeding areas . One
Arctic tern , ringed ( banded ) as a chick on the Farne Islands off the British east coast , reach
ed Melbourne , Australia in just three months from fledging , a sea journey of over 22 , 000 km (
14 , 000 mi ) . Many tubenosed birds breed in the southern hemisphere and migrate north in the sou
thern winter . ',
 b'The Arctic tern')
```

In [5]:

```
#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: Who was President of the United States prior to Eisenhower ?

context:  President Truman , symbolizing a broad - based desire for an Eisenhower candidacy for pr
esident , again in 1951 pressed him to run for the office as a Democrat . It was at this time that
Eisenhower voiced his disagreements with the Democratic party and declared himself and his family
to be Republicans . A " Draft Eisenhower " movement in the Republican Party persuaded him to decla
re his candidacy in the 1952 presidential election to counter the candidacy of non -
interventionist Senator Robert A . Taft . The effort was a long struggle ; Eisenhower had to be co
nvinced that political circumstances had created a genuine duty for him to offer himself as a cand
idate , and that there was a mandate from the populace for him to be their President . Henry Cabot
Lodge , who served as his campaign manager , and others succeeded in convincing him , and in June
1952 he resigned his command at NATO to campaign full - time . Eisenhower defeated Taft for the no
mination , having won critical delegate votes from Texas . Eisenhower ' s campaign was noted for t
he simple but effective slogan , " I Like Ike " . It was essential to his success that Eisenhower
express opposition to Roosevelt ' s policy at Yalta and against Truman ' s policies in Korea and C
hina — matters in which he had once participated . In defeating Taft for the nomination , it becam
e necessary for Eisenhower to appease the right wing Old Guard of the Republican Party ; his
selection of Richard M . Nixon as the Vice - President on the ticket was designed in part for that
purpose . Nixon also provided a strong anti - communist presence as well as some youth to counter
Ike ' s more advanced age .

target: Truman
```

2.2 Decoding from a fine-tuned model

You will now use an existing model that we trained for you. You will initialize, then load in your model, and then try with your own input.

In [6]:

```
# 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 [7]:

```
# load in the model
# this will take a minute
shape11 = trax.shapes.ShapeDtype((1, 1), dtype=np.int32)
model.init_from_file('model_squad.pkl.gz',
                     weights_only=True, input_signature=(shape11, shape11))
```

In [8]:

```
# 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 f
lowering plant of the genus Rosa, in the family Rosaceae, or the flower it bears.There are over th
ree hundred species and tens of thousands of cultivars. They form a group of plants that can be er
ect shrubs, climbing, or trailing, with stems that are often armed with sharp prickles. Flowers va
ry in size and shape and are usually large and showy, in colours ranging from white through yellow
s and reds. Most species are native to Asia, with smaller numbers native to Europe, North America,
and northwestern Africa. Species, cultivars and hybrids are all widely grown for their beauty and
often are fragrant.'
```

In [9]:

```
# 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]
```

Run the cell below to decode.

Note: This will take some time to run

In [10]:

```
# 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=5) # originally max_length=10
print(wrapper.fill(pretty_decode(output[0])))
```

```
white through yellows and
```

You should also be aware that the quality of the decoding is not very good because max_length was downsized from 10 to 5 so that this runs faster within this environment. The colab version uses the original max_length so check that one for the actual decoding.