Assignment 3 Ungraded Sections - Part 1: BERT Loss Model

Welcome to the part 1 of testing the models for this week's assignment. We will perform decoding using the BERT Loss model. In this notebook we'll use an input, mask (hide) random word(s) in it and see how well we get the "Target" answer(s).

Colab

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

BERT Loss 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 pickle
import string
import ast
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]:

```
example_jsons = list(map(ast.literal_eval, open('data.txt')))

natural_language_texts = [example_json['text'] for example_json in example_jsons]

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))


inputs_targets_pairs = []

# here you are reading already computed input/target pairs from a file
with open ('inputs_targets_pairs_file.txt', 'rb') as fp:
    inputs_targets_pairs = pickle.load(fp)


def display_input_target_pairs(inputs_targets_pairs):
    for i, inp_tgt_pair in enumerate(inputs_targets_pairs, 1):
        inps, tgts = inp_tgt_pair
        inps, tgts = pretty_decode(inps), pretty_decode(tgts)
        print(f'[{i}]\n'
              f'inputs:\n{wrapper.fill(text=inps)}\n\n'
              f'targets:\n{wrapper.fill(text=tgts)}\n\n\n\n')

display_input_target_pairs(inputs_targets_pairs)
```

```
[1]
inputs:
Beginners BBQ <Z> Taking <Y> in Missoula! <X> want to get better <W>
making delicious <V>? You will have the opportunity, put this on <U>
calendar now <T> Thursday, September 22nd<S> World Class BBQ Champion,
Tony Balay from Lonestar Smoke Rangers. He<R> be <Q> a beginner<P>
class for everyone <O> wants to<N> better <M> their <L> skills. He
will teach you <K> you need to know <J> compete in  <I> KCBS BBQ
competition, including techniques, recipes,<H>s, meat selection<G>
trimming, plus smoker <F> information. The cost to be in the class is
$35 per person<E> for spectator<D> is free. Included in the cost will
be either a t-shirt or apron and you will<C> tasting samples of each
```

be either a t-shirt or apron and you will<C> tasting samples of each meat that <B>.

targets:
<Z> Class <Y> Place <X> Do you <W> at <V> BBQ <U> your <T>.<S> join<R> will <Q> teaching<P> level <O> who<N> get <M> with <L> culinary <K> everything <J> to <I>a<H> timeline<G> and <F> and fire<E>, and<D>s it<C> be <B> is prepared

[2]
inputs:
Discussion <Z> ' <Y> X Lion (10.7)' started by <X>xboi87, Jan <W> 2012. I've got a 500gb <V> drive and <U> 240gb SSD <T> When trying to restore using disk utility i'm given the<S>Not enough space<R> disk ___ <Q> to restore<P> But I shouldn't have to do that!!! <O> or<N>s before resorting to the <M>? Use <L> Copy <K>ner to copy one drive to the other. <J>'ve done this several times <I> larger<H>D<G> smaller SSD and I wound up with a bootable SSD drive. One step you have to remember not to skip is <F> use Disk Utility to partition the SSD as GUID partition scheme HFS+<E> doing the<D>ne<C> If it came Apple Partition Scheme, even if you let CCC do the <B>e <A> the resulting <z> won' <y> be bootable<x> CCC<w> works in "file mode" and it can easily copy a larger drive (that's mostly empty) onto<v> drive.<u> you tell CCC to clone <t>a drive <s> did NOT boot from, it can work in<r> copy mode where the destination drive must be the same size or larger than<q> drive you are cloning from<p>if I recall). I've actually done <o> somehow <n> Disk Utility several times <m>booting from <l>a different drive (<k> even the dvd) so not running <j> utility from the drive your cloning) and had it<i> just fine from larger to smaller bootable clone. Definitely<h> drive <g>ning to first, as bootable Apple <f>.. Thanks for pointing this out<e> only experience using DU to go larger to smaller was when I was trying to make <d>a <c> install <b> I was unable to restore InstallESD.d <a>g Théâtre Keep 4 GB USB stick but of course the reason that wouldndürftigt fit isutti was slightly more than 4 Carolyn of data.

targets:
<Z> in <Y>Mac OS <X>a <W> 20, <V> internal <U>a <T>.<S> error "<R> on <Q>_<P>" <O> Any ideas<N> workaround <M> above <L> Carbon <K> Clo <J> I <I> going from<H> HD<G> to <F> to<E> before<D> clo<C>. <B>clon <A>, <z> drive <y>t<x>.<w> usually<v> a smaller<u> If <t> <s> you<r> block<q> the<p> (<o> this <n> on <m> (<l> <k>or <j> disk<i> work<h> format the<g>clo <f> etc<e>. My <d> <c> Lion <b> stick and <a>m Théâtre toKeepadürftig'utti there CarolynGB

[3]
inputs:
Fo <Z> plaid ly <Y> and <X>dex shortall with metallic slinky insets. Attached metallic elastic <W> with O-ring. <V>band <U>. Great hip hop <T> dance costume. Made in the USA.

targets:
<Z>il <Y>cra <X> span <W> belt <V> Head <U> included <T> or jazz

[4]
inputs:
How many back <Z>s per day for <Y> site? Discussion in 'Black Hat SEO <X> by Omoplat <W>, Dec 3, 2010. 1) <V> a newly created site, what's the max # back <U>s per day I should do to be <T>? 2) how long do I have to let my site age before I can start<S> blinks? I did about <R> profiles every 24 hours <Q> 10 days for one of<P> sites which had a brand new domain. There is three backlinks for <O> of these<N> profile so thats 18 000 backlinks <M> 24 hours and nothing happened in terms of being penalized or sandboxed. This is now <L> 3 <K> ago and the site is ranking on first page for a lot of my targeted keywords <J> build more you can in starting but <I> manual<H> and not spammy type<G> manual + relevant to the <F>.. then after 1 month you can make a big<E>.. Wow, dude, you built 18k backlinks a day on a brand new<D>? How quickly<C> rank up? What <B> of competition/searches did these

How quickly<C> rank up? What <B> of competition/searches did those
keywords have?

targets:
<Z>link <Y> new <X>' started <W>a <V> for <U>link <T> safe<S> making
more<R>6000 forum <Q> for<P> my <O> every<N> forum <M> every <L> maybe
<K> months <J>. <I> do<H> submission<G> means <F> post<E> blast<D>
site<C> did you <B> kind

[5]
inputs:
The Denver Board of Education opened the 2017-18 school year with an
update on projects that include new construction, upgrades, heat
mitigation and quality <Z>. We are excited <Y> students will be the
<X> of a four year, <W>72 million <V> Obligation Bond. Since the
passage of <U> bond, <T> construction team has worked to<S> the
projects over the four-year term of<R> bond. Denver <Q> on Tuesday
approved<P> and mill funding measures for students in <O> Public
Schools, agreeing to invest $5<N> million in bond funding to build and
improve schools and $5 <M> million in <L> dollars to support proven
initiatives, such as early literacy. Denver voters say yes to bond and
mill levy funding support <K> DPS students and schools. Click to learn
<J> about the details <I> voter-approved bond measure. Denver voters
on Nov. 8 approved bond and mill funding measures for DPS students and
schools. Learn<H> about what's included<G> the mill levy measure.

targets:
<Z> learning environments <Y> that Denver <X> beneficiaries <W> $5 <V>
General <U> the <T> our<S> schedule<R> the <Q> voters<P> bond <O>
Denver<N>72 <M>6.6 <L> operating <K> for <J> more <I> of the<H>
more<G> in

Part 2: BERT Loss

Now that you created the encoder, we will not make you train it. Training it could easily cost you a few days depending on which GPUs/TPUs you are using. Very few people train the full transformer from scratch. Instead, what the majority of people do, they load in a pretrained model, and they fine tune it on a specific task. That is exactly what you are about to do. Let's start by initializing and then loading in the model.

Initialize the model from the saved checkpoint.

In [3]:

```
# Initializing 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')
```

In [4]:

```
# Now load in the model
# this takes about 1 minute
shape11 = trax.shapes.ShapeDtype((1, 1), dtype=np.int32)  # Needed in predict mode.
model.init_from_file('model.pkl.gz',
                     weights_only=True, input_signature=(shape11, shape11))
```

2.1 Decoding

Now you will use one of the `inputs_targets_pairs` for input and as target. Next you will use the `pretty_decode` to output the input and target. The code to perform all of this has been provided below.

In [5]:

```
# using the 3rd example
c4_input = inputs_targets_pairs[2][0]
c4_target = inputs_targets_pairs[2][1]

print('pretty_decoded input: \n\n', pretty_decode(c4_input))
print('\npretty_decoded target: \n\n', pretty_decode(c4_target))
print('\nc4_input:\n\n', c4_input)
print('\nc4_target:\n\n', c4_target)
print(len(c4_target))
print(len(pretty_decode(c4_target)))
```

```
pretty_decoded input:

 Fo <Z> plaid ly <Y> and <X>dex shortall with metallic slinky insets. Attached metallic elastic <W
> with O-ring. <V>band <U>. Great hip hop <T> dance costume. Made in the USA.

pretty_decoded target:

 <Z>il <Y>cra <X> span <W> belt <V> Head <U> included <T> or jazz

c4_input:

 [4452, 31999, 30772, 3, 120, 31998, 11, 31997, 26, 994, 710, 1748, 28, 18813, 3, 7, 4907, 63, 16,
2244, 7, 5, 28416, 15, 26, 18813, 15855, 31996, 28, 411, 18, 1007, 5, 31995, 3348, 31994, 5, 1651,
5436, 13652, 31993, 2595, 11594, 5, 6465, 16, 8, 2312, 5]

c4_target:

 [31999, 173, 31998, 2935, 31997, 8438, 31996, 6782, 31995, 3642, 31994, 1285, 31993, 42, 9948]
15
64
```

Run the cell below to decode.

Note: This will take some time to run

In [6]:

```
# 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(c4_input)[None, :],
                                        temperature=0.0, max_length=5) # originally max_length = 50
print(wrapper.fill(pretty_decode(output[0])))
```

```
<Z>o <Y>cra <X>
```

At this point the RAM is almost full, this happens 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 an answer that makes no sense, or repetitive words. If that happens restart the runtime (see how to at the start of the notebook) and run all the cells again.

You should also be aware that the quality of the decoding is not very good because max_length was downsized from 50 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.