Assignment 3, 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).

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) 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 [1]: !pip -q install trax==1.3.4
```

```
     |████████████████████████████████| 368kB 4.4MB/s
     |████████████████████████████████| 163kB 14.1MB/s
     |████████████████████████████████| 2.6MB 18.9MB/s
     |████████████████████████████████| 1.5MB 42.0MB/s
     |████████████████████████████████| 1.0MB 59.8MB/s
     |████████████████████████████████| 3.5MB 54.6MB/s
     |████████████████████████████████| 1.1MB 55.1MB/s
     |████████████████████████████████| 307kB 50.2MB/s
     |████████████████████████████████| 71kB 11.3MB/s
     |████████████████████████████████| 655kB 57.9MB/s
     |████████████████████████████████| 194kB 54.8MB/s
     |████████████████████████████████| 368kB 55.1MB/s
     |████████████████████████████████| 5.3MB 55.3MB/s
     |████████████████████████████████| 983kB 45.9MB/s
     |████████████████████████████████| 358kB 57.2MB/s
     |████████████████████████████████| 81kB 12.7MB/s
     |████████████████████████████████| 3.0MB 54.8MB/s
     |████████████████████████████████| 890kB 56.0MB/s
     |████████████████████████████████| 235kB 59.7MB/s
     |████████████████████████████████| 51kB 7.4MB/s
  Building wheel for pypng (setup.py) ... done
  Building wheel for bz2file (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 [2]:
```python
import pickle
import string
import ast
import numpy as np
import trax
from trax.supervised import decoding
import textwrap
# Will come handy later.
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
```

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 [3]:
```python
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 [4]:
```python
example_jsons = list(map(ast.literal_eval, open("/content/drive/My Drive/NLP C4 W3 Data/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='/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))


inputs_targets_pairs = []

# here you are reading already computed input/target pairs from a file
with open ('/content/drive/My Drive/NLP C4 W3 Data/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')
```

In [6]:
```
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
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 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 3: BERT Loss

We will not train the encoder which you have built in the assignment (coursera version). 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 [5]:
```
# 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')  # Change to 'eval' for slow decoding.
```

In [7]:
```
# 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('/content/drive/My Drive/NLP C4 W3 Data/models/model.pkl.gz',
                     weights_only=True, input_signature=(shape11, shape11))
```

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

3.1 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 [9]:
```
# # using the 3rd example
# c4_input = inputs_targets_pairs[2][0]
# c4_target = inputs_targets_pairs[2][1]

# using the 1st example
c4_input = inputs_targets_pairs[0][0]
c4_target = inputs_targets_pairs[0][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:

 Beginners BBQ <Z> Taking <Y> in Missoula! <X> want to get better <W> making delicious <V>? You will have the opportun
ity, put this on <U> calendar now <T> Thursday, September 22nd<S> World Class BBQ Champion, Tony Balay from Lonestar S
moke 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. I
ncluded in the cost will be either a t-shirt or apron and you will<C> tasting samples of each meat that <B>.

pretty_decoded target:

 <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

c4_input:

 [12847, 277, 15068, 31999, 3, 12297, 31998, 16, 5964, 7115, 9, 55, 31997, 241, 12, 129, 394, 31996, 492, 3326, 31995,
58, 148, 56, 43, 8, 1004, 6, 474, 48, 30, 31994, 4793, 230, 31993, 2721, 6, 1600, 1630, 727, 31992, 1150, 4501, 15068,
16127, 6, 9137, 2659, 5595, 45, 301, 782, 3624, 14627, 15, 12612, 277, 5, 216, 31991, 36, 31990, 3, 9, 19529, 31989, 8
53, 21, 921, 31988, 2746, 12, 31987, 394, 31986, 70, 31985, 1098, 5, 216, 56, 3884, 25, 31984, 25, 174, 12, 214, 3198
3, 5978, 16, 3, 31982, 3, 23405, 4547, 15068, 2259, 6, 379, 2097, 6, 5459, 6, 31981, 7, 6, 3604, 1801, 31980, 27856,
6, 303, 24190, 31979, 251, 5, 37, 583, 12, 36, 16, 8, 853, 19, 25264, 399, 568, 31978, 21, 21380, 31977, 19, 339, 5, 1
5746, 26, 16, 8, 583, 56, 36, 893, 3, 9, 3, 17, 18, 9486, 42, 3, 9, 1409, 29, 11, 25, 56, 31976, 12246, 5977, 13, 284,
3604, 24, 31975, 5]

c4_target:

 [31999, 4501, 31998, 3399, 31997, 531, 25, 31996, 44, 31995, 15068, 31994, 39, 31993, 5, 31992, 1715, 31991, 56, 3199
0, 2119, 31989, 593, 31988, 113, 31987, 129, 31986, 28, 31985, 17712, 31984, 762, 31983, 12, 31982, 9, 31981, 13618, 3
1980, 11, 31979, 11, 1472, 31978, 6, 11, 31977, 7, 34, 31976, 36, 31975, 19, 2657]
55
230
```

Run the cell below to decode

In [10]:
```
# Faster decoding: (still - maybe lower max_length to 20 for speed)
# 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=50)
print(wrapper.fill(pretty_decode(output[0])))
```

```
<Z> Class <Y> Place <X> Do you <W> at <V> BBQ <U> your <T>! This<S>,
is the first class taught by<R> will <Q> teaching<P> BBQ <O> who<N>
get <M> at <L> BBQ <K> everything <J> to <I>a<H> judging, judging
```

Note: As you can see the RAM is almost full, it is because the model and the decoding is memory heavy. Running it the second time 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.

In [ ]:
```
```