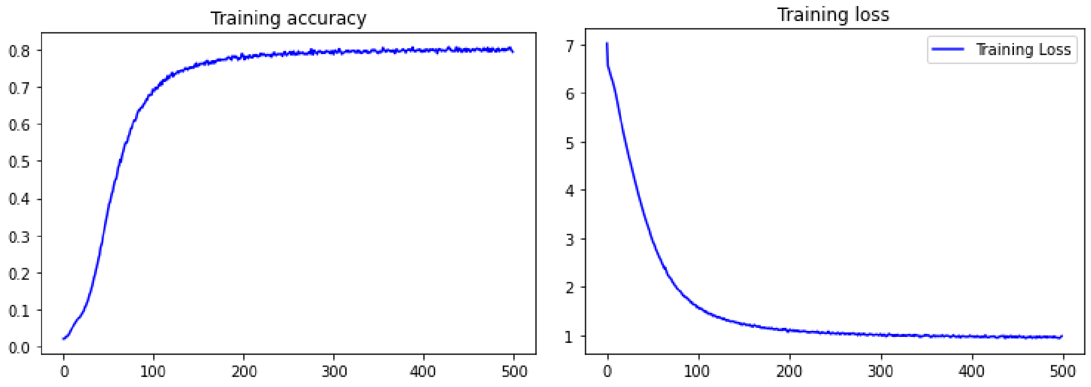
Neural Text Generation

- ***Introduction:***
  In the past few years, various advancements have been made in Language Models owing to the formulation of various new algorithms such as Generative Adversarial Networks (GANs), ELMo and Bidirectional Encoder Representations from Transformers (BERT). Text Generation, one of the most important Language Modelling problems, has shown great promise recently due to the advancement of more efficient and competent context dependent algorithms such as BERT and GPT-2 compared to preceding context independent algorithms such as word2vec and GloVe.

- ***Approach:***

1. **Model from scratch**

   Before directly moving to GPT2, or Textgenrnn, I tried to generate independent text samples using Bidirectional LSTMs. After testing the model on a sample dataset, following is the graphs for acc and loss:



Given the following seed text

```
seed_text = "Help me Obi Wan Kenobi, you're my only hope"
next_words = 100
```

Here is the next 100 words generated for the seed text

```
Help me Obi Wan Kenobi, you're my only hope to your fame canst live hold
aside made so bright stay dead old near or night and cheeks gather'd
express'd now give another part remember upon my beck fall lies are mine ' '
' ' alone doth dwell are seem so stout are taken new prime suppose woe ' ' '
' ' ' or mine own love to my heart still tend depart aside live in thine
dyed eyes live on ill eyes crush'd on eyes survey are new fire held fire
outworn stand bear another prime hour wilfully invocate invocate now deceive
ill gone dead esteem'd or
```

The text samples seem pretty good.

I finally moved to building an actual production level model.
I used a custom wikipedia sentences dataset.
Link for the data: https://www.kaggle.com/mikeortman/wikipedia-sentences

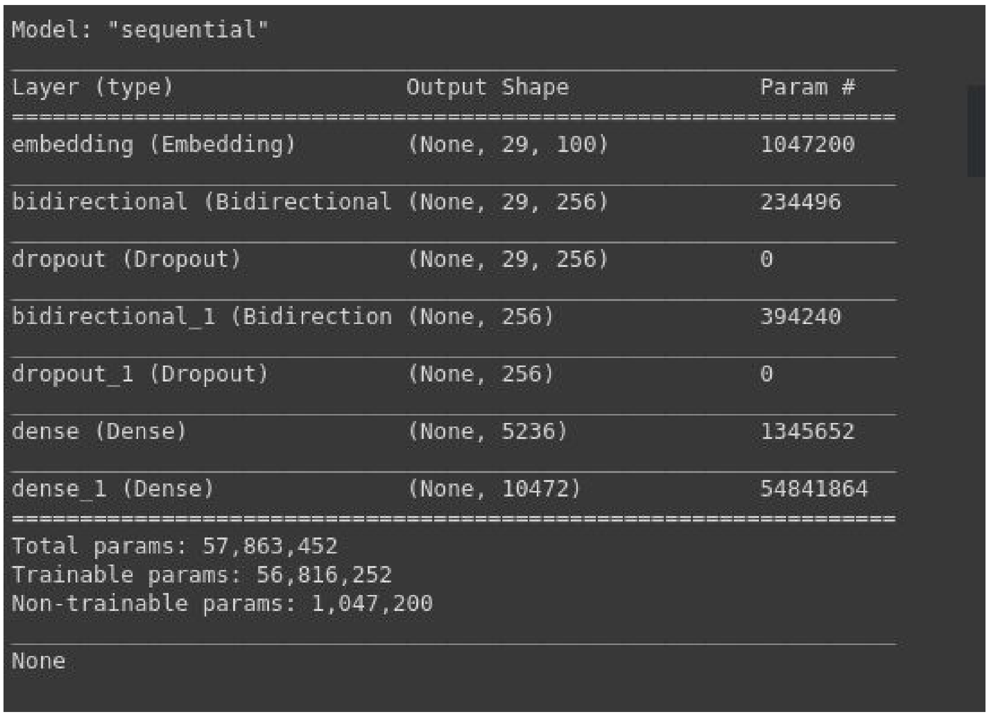
It is a collection of 7.8 million sentences (one per line) from August 2018 English wikipedia. These are only sentences found in the opening text of content pages. Further, filtering is applied to remove junk sentences.

In Deep Learning, ‘Data is power’. Having received data of 7.8 million lines, I was super excited to build a production level text classification model.

But I realized that handling this data is out of scope for the 12GB RAM of Colab too XD. So I reduced the data to a size which is large enough to juice out all the power which colab provides

I have used the 100 D GloVe embedding for the Kera Embedding layer

Here is the model architecture which I used



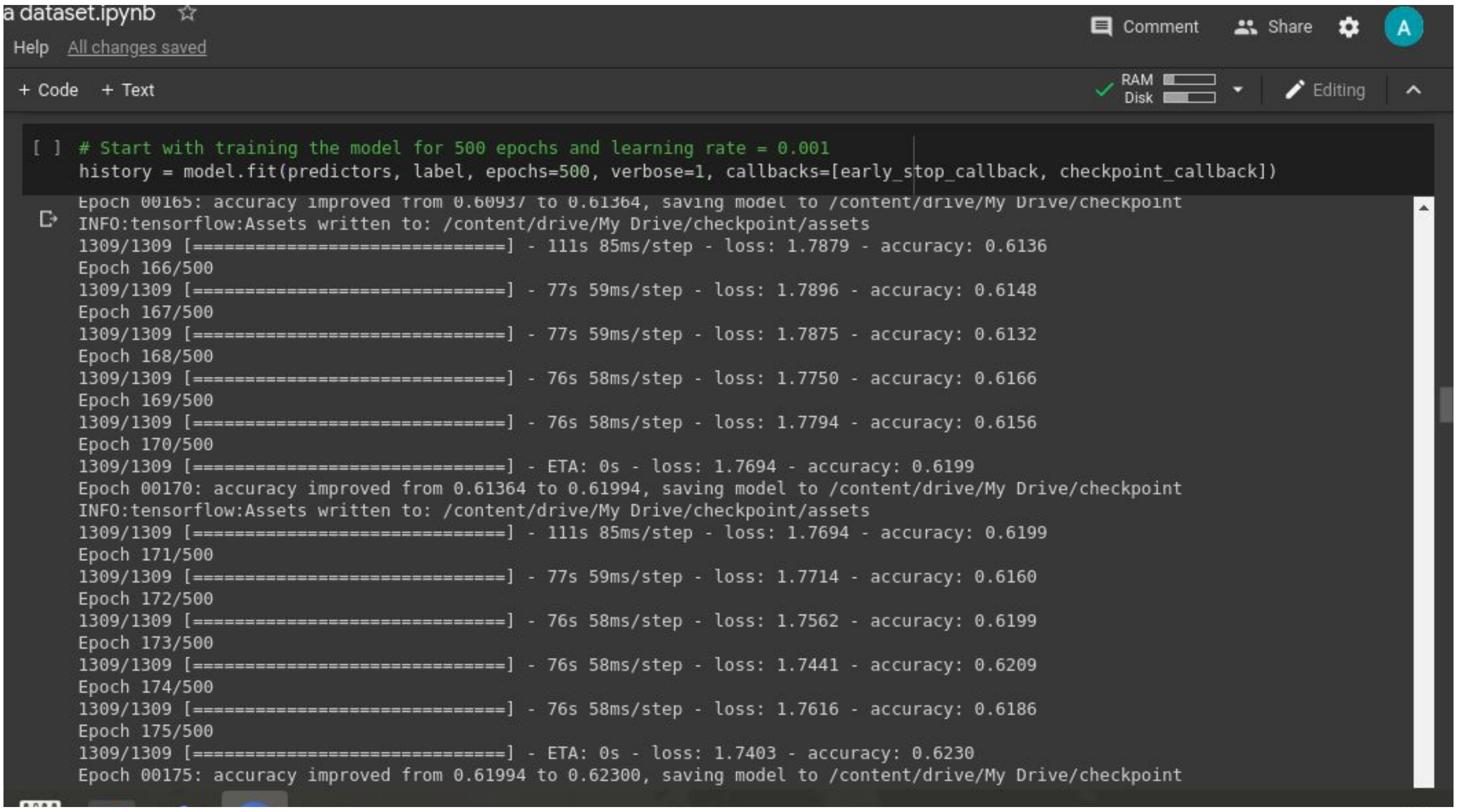
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 29, 100)           1047200
_________________________________________________________________
bidirectional (Bidirectional (None, 29, 256)           234496
_________________________________________________________________
dropout (Dropout)            (None, 29, 256)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               394240
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 5236)              1345652
_________________________________________________________________
dense_1 (Dense)              (None, 10472)             54841864
=================================================================
Total params: 57,863,452
Trainable params: 56,816,252
Non-trainable params: 1,047,200
_________________________________________________________________
None
```

Then we start training the model. I used following callbacks:

```
# Callbacks for stopping the model early if loss does not reduce for 4 successive
epochs (patience = 4)
early_stop_callback = tf.keras.callbacks.EarlyStopping(
   monitor='loss', min_delta=0, patience=4, verbose=1, mode='auto',
   baseline=None, restore_best_weights=False)
```

```
# Callback for saving the complete model (.pb file) after every 5th epoch

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
   filepath=checkpoint_filepath,
   save_weights_only=False,
   monitor='accuracy',
   verbose = 1,
   mode='auto',
   period = 5,
   save_best_only=True)
```

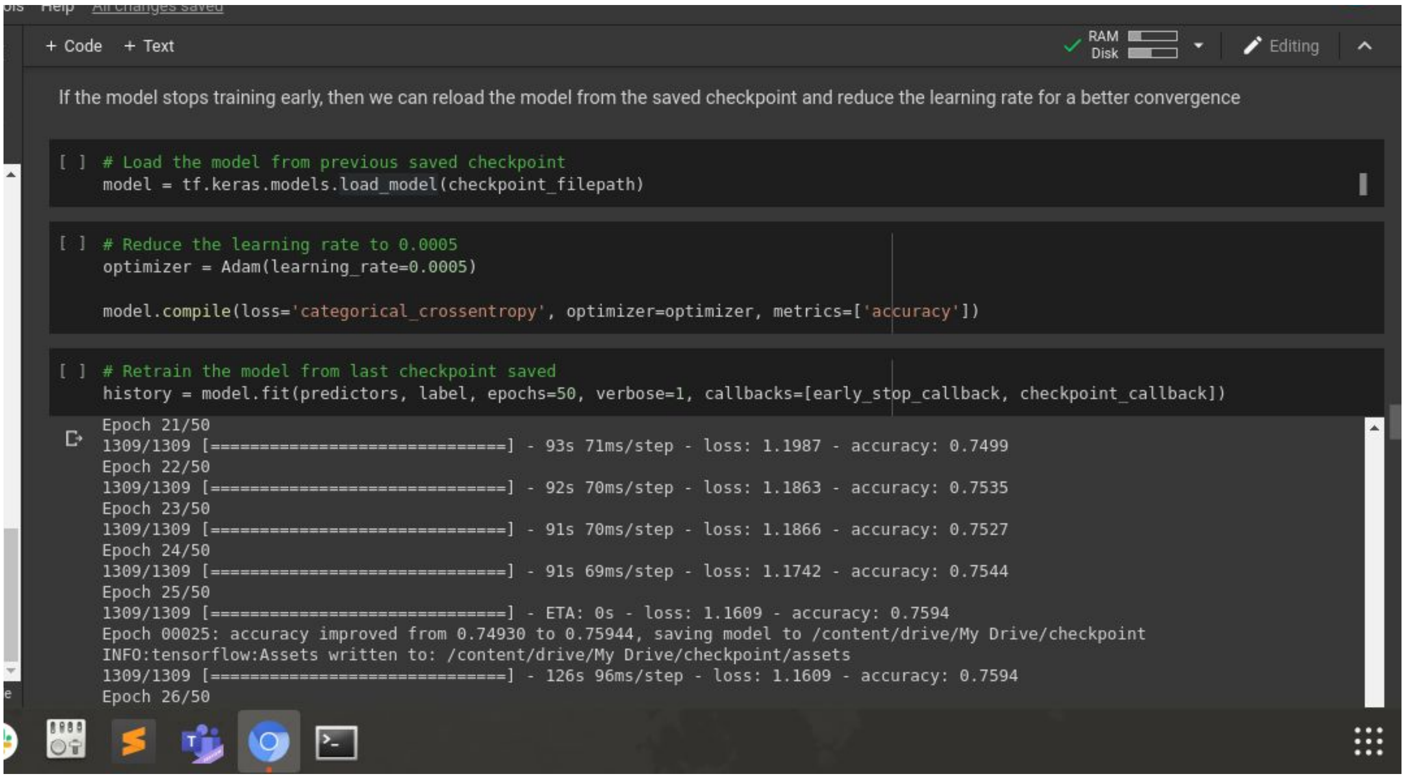
Model training:



```
[ ] # Start with training the model for 500 epochs and learning rate = 0.001
    history = model.fit(predictors, label, epochs=500, verbose=1, callbacks=[early_stop_callback, checkpoint_callback])
```

```
Epoch 00165: accuracy improved from 0.60937 to 0.61364, saving model to /content/drive/My Drive/checkpoint
INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
1309/1309 [==============================] - 111s 85ms/step - loss: 1.7879 - accuracy: 0.6136
Epoch 166/500
1309/1309 [==============================] - 77s 59ms/step - loss: 1.7896 - accuracy: 0.6148
Epoch 167/500
1309/1309 [==============================] - 77s 59ms/step - loss: 1.7875 - accuracy: 0.6132
Epoch 168/500
1309/1309 [==============================] - 76s 58ms/step - loss: 1.7750 - accuracy: 0.6166
Epoch 169/500
1309/1309 [==============================] - 76s 58ms/step - loss: 1.7794 - accuracy: 0.6156
Epoch 170/500
1309/1309 [==============================] - ETA: 0s - loss: 1.7694 - accuracy: 0.6199
Epoch 00170: accuracy improved from 0.61364 to 0.61994, saving model to /content/drive/My Drive/checkpoint
INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
1309/1309 [==============================] - 111s 85ms/step - loss: 1.7694 - accuracy: 0.6199
Epoch 171/500
1309/1309 [==============================] - 77s 59ms/step - loss: 1.7714 - accuracy: 0.6160
Epoch 172/500
1309/1309 [==============================] - 76s 58ms/step - loss: 1.7562 - accuracy: 0.6199
Epoch 173/500
1309/1309 [==============================] - 76s 58ms/step - loss: 1.7441 - accuracy: 0.6209
Epoch 174/500
1309/1309 [==============================] - 76s 58ms/step - loss: 1.7616 - accuracy: 0.6186
Epoch 175/500
1309/1309 [==============================] - ETA: 0s - loss: 1.7403 - accuracy: 0.6230
Epoch 00175: accuracy improved from 0.61994 to 0.62300, saving model to /content/drive/My Drive/checkpoint
```

I have used start and stop training method for a better convergence
Once the model stops, then we load it again from the previous saved checkpoint.
Then we retrain the model with a lesse learning rate. Reducing the learning rate by some amount (advisably 1/10th, or 1/5th) helps model to reach a better convergence and improves accuracy



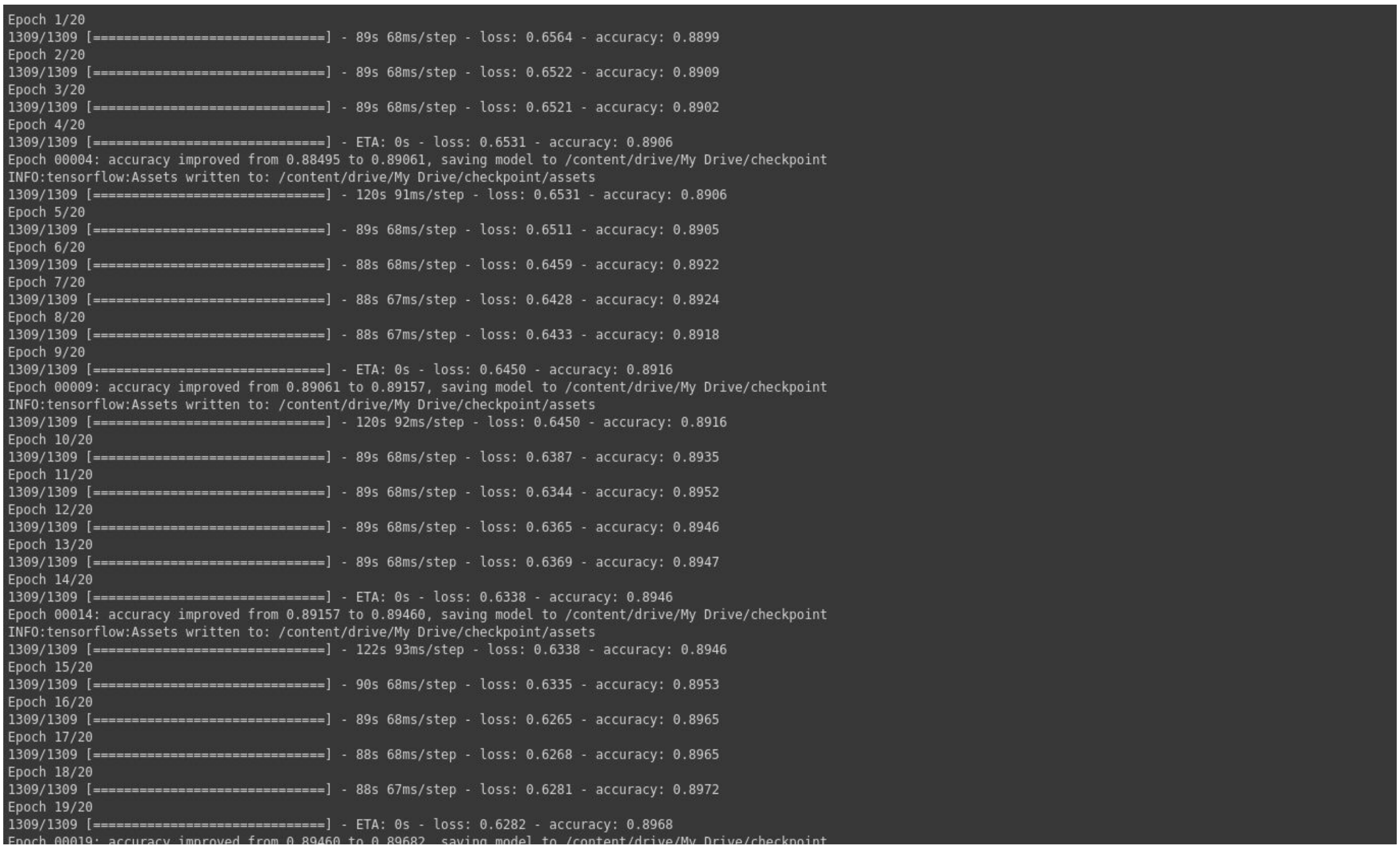
If the model stops training early, then we can reload the model from the saved checkpoint and reduce the learning rate for a better convergence

```
[ ] # Load the model from previous saved checkpoint
    model = tf.keras.models.load_model(checkpoint_filepath)
```

```
[ ] # Reduce the learning rate to 0.0005
    optimizer = Adam(learning_rate=0.0005)

    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

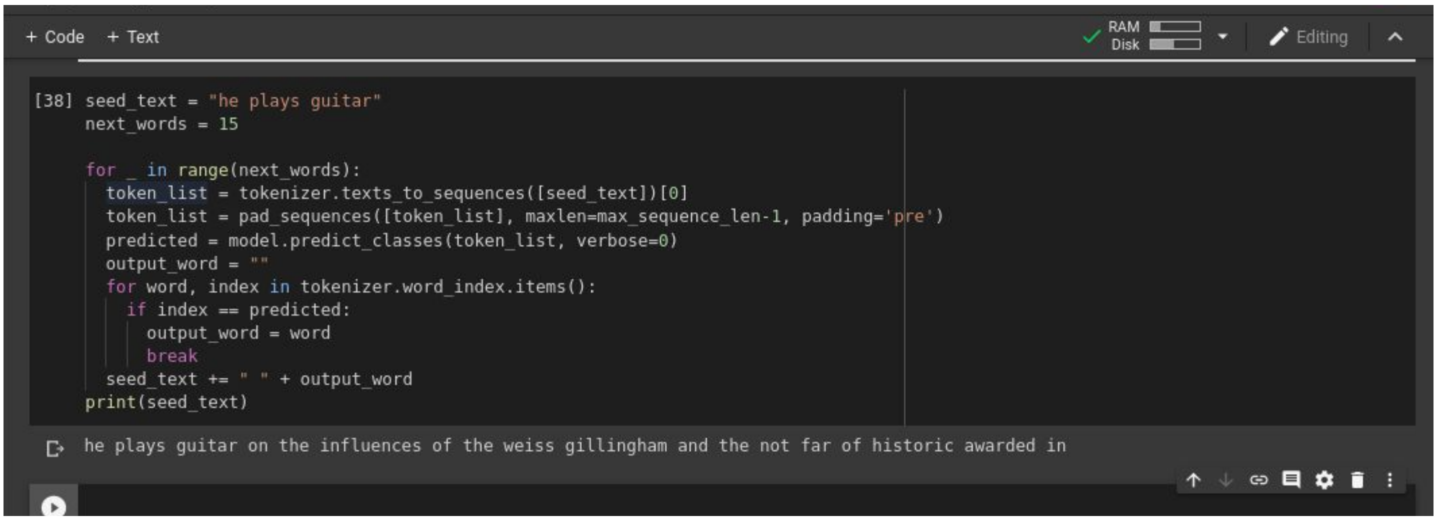
```
[ ] # Retrain the model from last checkpoint saved
    history = model.fit(predictors, label, epochs=50, verbose=1, callbacks=[early_stop_callback, checkpoint_callback])

    Epoch 21/50
    1309/1309 [==============================] - 93s 71ms/step - loss: 1.1987 - accuracy: 0.7499
    Epoch 22/50
    1309/1309 [==============================] - 92s 70ms/step - loss: 1.1863 - accuracy: 0.7535
    Epoch 23/50
    1309/1309 [==============================] - 91s 70ms/step - loss: 1.1866 - accuracy: 0.7527
    Epoch 24/50
    1309/1309 [==============================] - 91s 69ms/step - loss: 1.1742 - accuracy: 0.7544
    Epoch 25/50
    1309/1309 [==============================] - ETA: 0s - loss: 1.1609 - accuracy: 0.7594
    Epoch 00025: accuracy improved from 0.74930 to 0.75944, saving model to /content/drive/My Drive/checkpoint
    INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
    1309/1309 [==============================] - 126s 96ms/step - loss: 1.1609 - accuracy: 0.7594
    Epoch 26/50
```

Using this method, the model was able to achieve accuracy of 89%



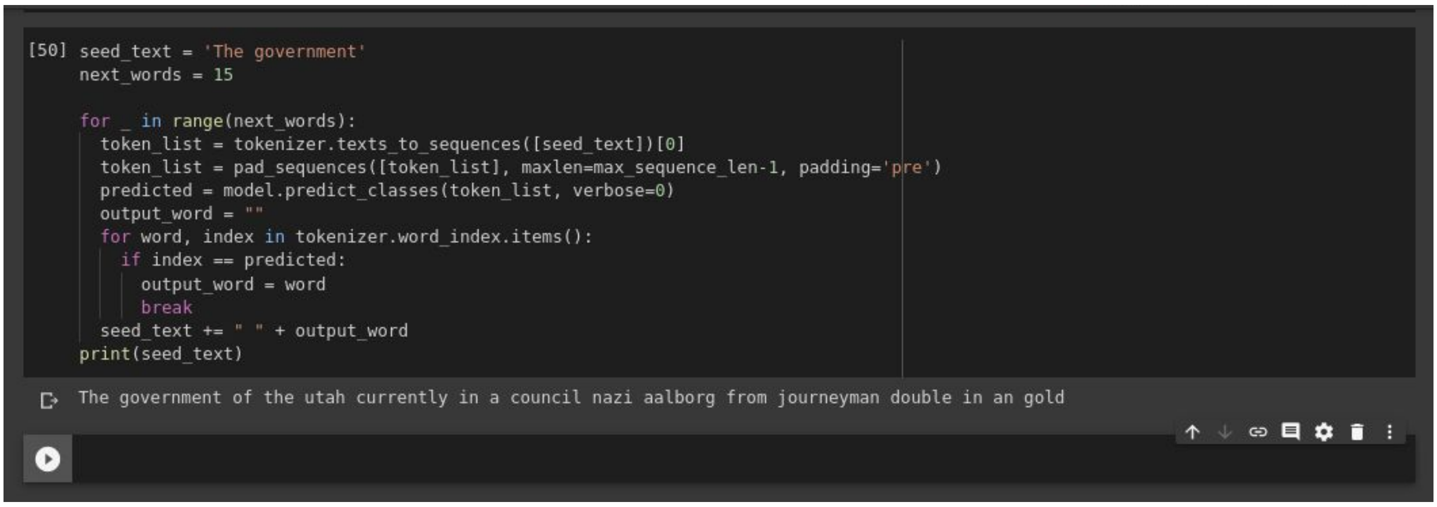
```
Epoch 1/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6564 - accuracy: 0.8899
Epoch 2/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6522 - accuracy: 0.8909
Epoch 3/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6521 - accuracy: 0.8902
Epoch 4/20
1309/1309 [==============================] - ETA: 0s - loss: 0.6531 - accuracy: 0.8906
Epoch 00004: accuracy improved from 0.88495 to 0.89061, saving model to /content/drive/My Drive/checkpoint
INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
1309/1309 [==============================] - 120s 91ms/step - loss: 0.6531 - accuracy: 0.8906
Epoch 5/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6511 - accuracy: 0.8905
Epoch 6/20
1309/1309 [==============================] - 88s 68ms/step - loss: 0.6459 - accuracy: 0.8922
Epoch 7/20
1309/1309 [==============================] - 88s 67ms/step - loss: 0.6428 - accuracy: 0.8924
Epoch 8/20
1309/1309 [==============================] - 88s 67ms/step - loss: 0.6433 - accuracy: 0.8918
Epoch 9/20
1309/1309 [==============================] - ETA: 0s - loss: 0.6450 - accuracy: 0.8916
Epoch 00009: accuracy improved from 0.89061 to 0.89157, saving model to /content/drive/My Drive/checkpoint
INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
1309/1309 [==============================] - 120s 92ms/step - loss: 0.6450 - accuracy: 0.8916
Epoch 10/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6387 - accuracy: 0.8935
Epoch 11/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6344 - accuracy: 0.8952
Epoch 12/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6365 - accuracy: 0.8946
Epoch 13/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6369 - accuracy: 0.8947
Epoch 14/20
1309/1309 [==============================] - ETA: 0s - loss: 0.6338 - accuracy: 0.8946
Epoch 00014: accuracy improved from 0.89157 to 0.89460, saving model to /content/drive/My Drive/checkpoint
INFO:tensorflow:Assets written to: /content/drive/My Drive/checkpoint/assets
1309/1309 [==============================] - 122s 93ms/step - loss: 0.6338 - accuracy: 0.8946
Epoch 15/20
1309/1309 [==============================] - 90s 68ms/step - loss: 0.6335 - accuracy: 0.8953
Epoch 16/20
1309/1309 [==============================] - 89s 68ms/step - loss: 0.6265 - accuracy: 0.8965
Epoch 17/20
1309/1309 [==============================] - 88s 68ms/step - loss: 0.6268 - accuracy: 0.8965
Epoch 18/20
1309/1309 [==============================] - 88s 67ms/step - loss: 0.6281 - accuracy: 0.8972
Epoch 19/20
1309/1309 [==============================] - ETA: 0s - loss: 0.6282 - accuracy: 0.8968
Epoch 00019: accuracy improved from 0.89460 to 0.89682, saving model to /content/drive/My Drive/checkpoint
```

Sample text generated



```
[38] seed_text = "he plays guitar"
     next_words = 15

     for _ in range(next_words):
       token_list = tokenizer.texts_to_sequences([seed_text])[0]
       token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
       predicted = model.predict_classes(token_list, verbose=0)
       output_word = ""
       for word, index in tokenizer.word_index.items():
         if index == predicted:
           output_word = word
           break
       seed_text += " " + output_word
     print(seed_text)
```

```
he plays guitar on the influences of the weiss gillingham and the not far of historic awarded in
```



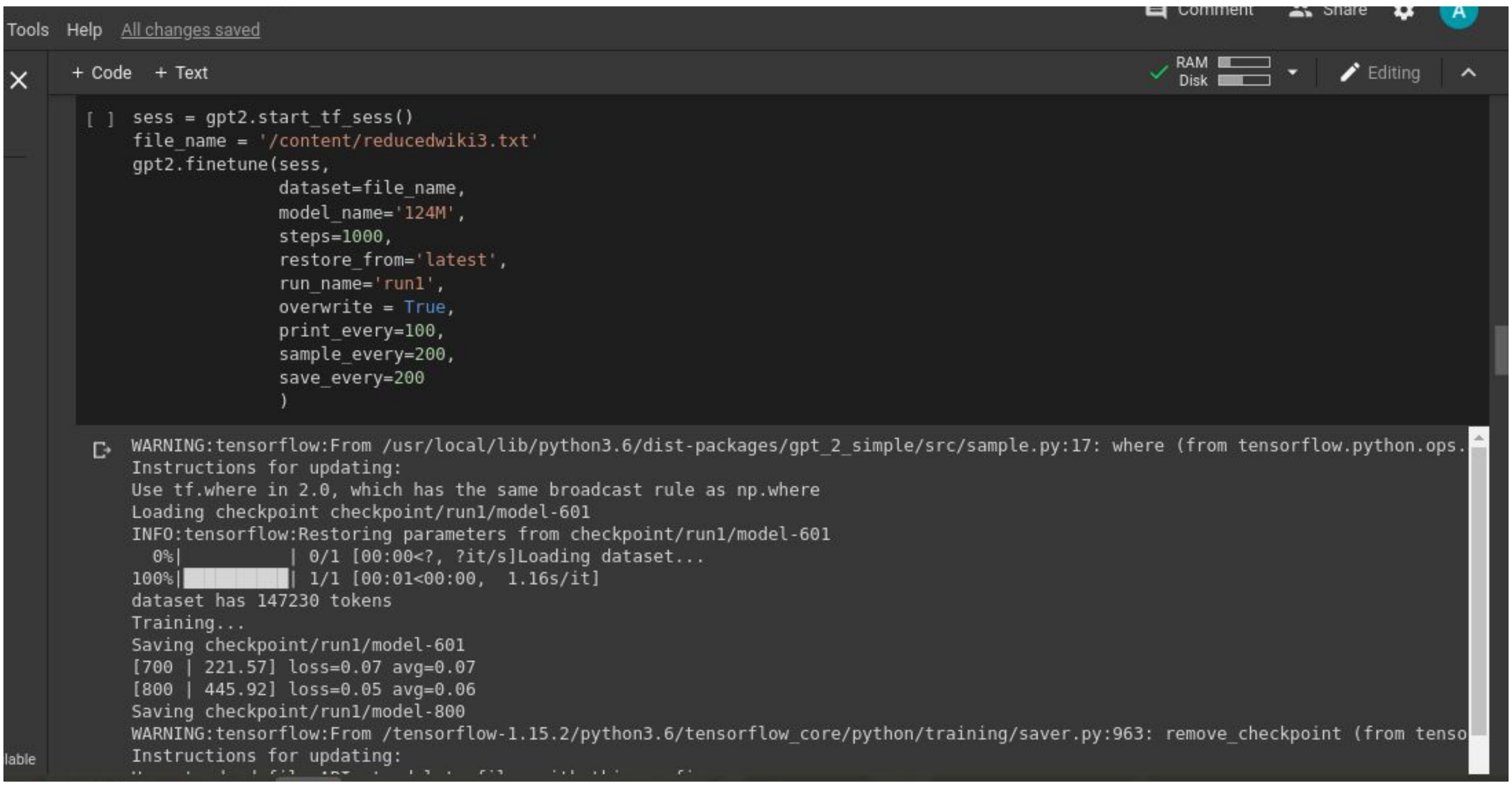
```
[50] seed_text = 'The government'
     next_words = 15

     for _ in range(next_words):
       token_list = tokenizer.texts_to_sequences([seed_text])[0]
       token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
       predicted = model.predict_classes(token_list, verbose=0)
       output_word = ""
       for word, index in tokenizer.word_index.items():
         if index == predicted:
           output_word = word
           break
       seed_text += " " + output_word
     print(seed_text)
```

```
The government of the utah currently in a council nazi aalborg from journeyman double in an gold
```

There’s a huge scope for improvement in generated text.
For eg: Instead of using greedy approach (selecting the word with best probability), a method called beam search can be used (selecting top k words where k = [2, 10]).
Other methods could be using ELMo embedding, a better model architecture, using transformers etc.

2. **Using the beast - GPT2**
   In February 2019, OpenAI released a paper describing GPT-2, a AI-based text-generation model based on the Transformer architecture and trained on massive amounts of text all around the internet.
   OpenAI has released three flavors of GPT-2 models to date: the “small” 124M parameter model (500MB on disk), the “medium” 355M model (1.5GB on disk), and recently the 774M model (3GB on disk). These models are much larger than what you see in typical AI tutorials and are harder to wield: the “small” model hits GPU memory limits while fine tuning with consumer GPUs, the “medium” model requires additional training techniques before it could be finetuned on server GPUs without going out-of-memory, and the “large” model cannot be finetuned at all with current server GPUs before going OOM, even with those techniques.
   Here, I used the ‘small’ 124M parameter model and fine tuned it on the same wikipedia sentences data.

   GPT2 fine tuned:

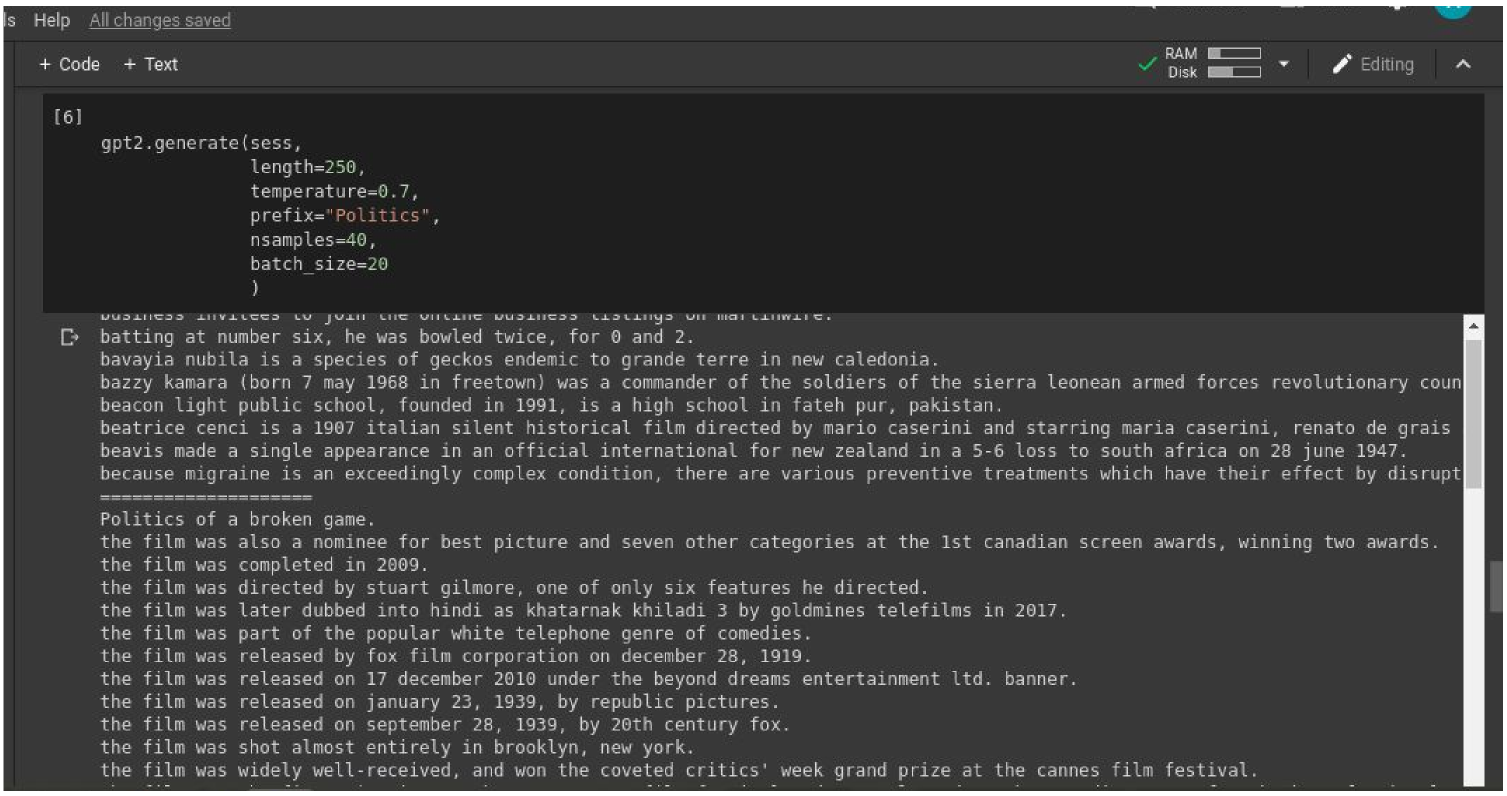


```
sess = gpt2.start_tf_sess()
file_name = '/content/reducedwiki3.txt'
gpt2.finetune(sess,
              dataset=file_name,
              model_name='124M',
              steps=1000,
              restore_from='latest',
              run_name='run1',
              overwrite = True,
              print_every=100,
              sample_every=200,
              save_every=200
              )
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/gpt_2_simple/src/sample.py:17: where (from tensorflow.python.ops.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Loading checkpoint checkpoint/run1/model-601
INFO:tensorflow:Restoring parameters from checkpoint/run1/model-601
  0%|          | 0/1 [00:00<?, ?it/s]Loading dataset...
100%|██████████| 1/1 [00:01<00:00,  1.16s/it]
dataset has 147230 tokens
Training...
Saving checkpoint/run1/model-601
[700 | 221.57] loss=0.07 avg=0.07
[800 | 445.92] loss=0.05 avg=0.06
Saving checkpoint/run1/model-800
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/training/saver.py:963: remove_checkpoint (from tenso
Instructions for updating:
```

   Once we fine tuned the model, then we can load the GPT model from the saved checkpoint and then use it for generating our own custom text

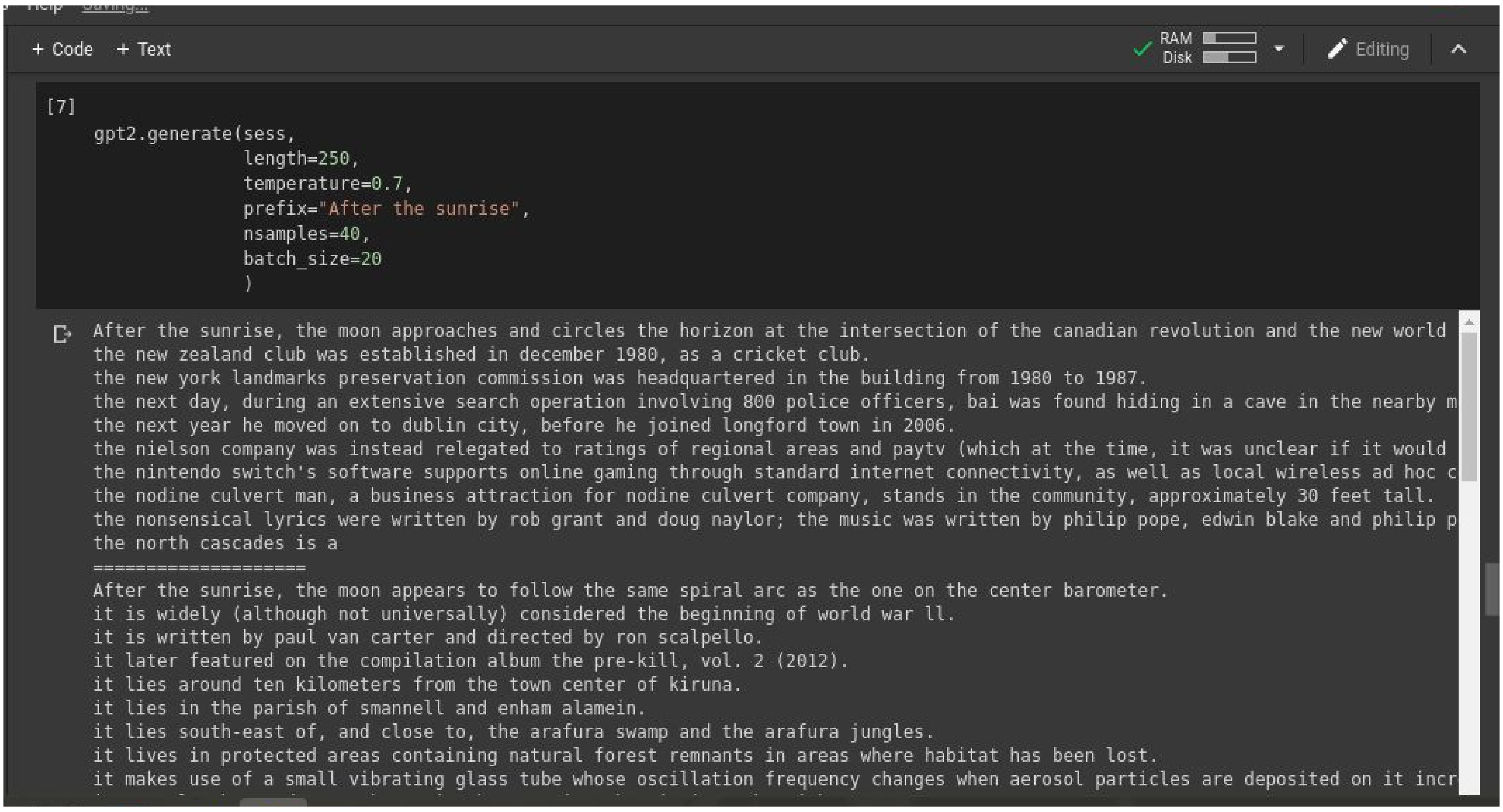
Text generated using the prefix “Politics”



```
[6]
    gpt2.generate(sess,
                  length=250,
                  temperature=0.7,
                  prefix="Politics",
                  nsamples=40,
                  batch_size=20
                  )
```

```
batting at number six, he was bowled twice, for 0 and 2.
bavayia nubila is a species of geckos endemic to grande terre in new caledonia.
bazzy kamara (born 7 may 1968 in freetown) was a commander of the soldiers of the sierra leonean armed forces revolutionary coun
beacon light public school, founded in 1991, is a high school in fateh pur, pakistan.
beatrice cenci is a 1907 italian silent historical film directed by mario caserini and starring maria caserini, renato de grais
beavis made a single appearance in an official international for new zealand in a 5-6 loss to south africa on 28 june 1947.
because migraine is an exceedingly complex condition, there are various preventive treatments which have their effect by disrupt
====================
Politics of a broken game.
the film was also a nominee for best picture and seven other categories at the 1st canadian screen awards, winning two awards.
the film was completed in 2009.
the film was directed by stuart gilmore, one of only six features he directed.
the film was later dubbed into hindi as khatarnak khiladi 3 by goldmines telefilms in 2017.
the film was part of the popular white telephone genre of comedies.
the film was released by fox film corporation on december 28, 1919.
the film was released on 17 december 2010 under the beyond dreams entertainment ltd. banner.
the film was released on january 23, 1939, by republic pictures.
the film was released on september 28, 1939, by 20th century fox.
the film was shot almost entirely in brooklyn, new york.
the film was widely well-received, and won the coveted critics' week grand prize at the cannes film festival.
```

Text generated for prefix “After the sunrise”



```
[7]
    gpt2.generate(sess,
                  length=250,
                  temperature=0.7,
                  prefix="After the sunrise",
                  nsamples=40,
                  batch_size=20
                  )
```

```
After the sunrise, the moon approaches and circles the horizon at the intersection of the canadian revolution and the new world
the new zealand club was established in december 1980, as a cricket club.
the new york landmarks preservation commission was headquartered in the building from 1980 to 1987.
the next day, during an extensive search operation involving 800 police officers, bai was found hiding in a cave in the nearby m
the next year he moved on to dublin city, before he joined longford town in 2006.
the nielson company was instead relegated to ratings of regional areas and paytv (which at the time, it was unclear if it would
the nintendo switch's software supports online gaming through standard internet connectivity, as well as local wireless ad hoc c
the nodine culvert man, a business attraction for nodine culvert company, stands in the community, approximately 30 feet tall.
the nonsensical lyrics were written by rob grant and doug naylor; the music was written by philip pope, edwin blake and philip p
the north cascades is a
====================
After the sunrise, the moon appears to follow the same spiral arc as the one on the center barometer.
it is widely (although not universally) considered the beginning of world war ll.
it is written by paul van carter and directed by ron scalpello.
it later featured on the compilation album the pre-kill, vol. 2 (2012).
it lies around ten kilometers from the town center of kiruna.
it lies in the parish of smannell and enham alamein.
it lies south-east of, and close to, the arafura swamp and the arafura jungles.
it lives in protected areas containing natural forest remnants in areas where habitat has been lost.
it makes use of a small vibrating glass tube whose oscillation frequency changes when aerosol particles are deposited on it incr
```

References:

https://arxiv.org/pdf/1711.09534.pdf
https://minimaxir.com/2019/09/howto-gpt2/
https://towardsdatascience.com/boosting-your-sequence-generation-performance-with-beam-search-language-model-decoding-74ee64de435a